# 4D RECONSTRUCTION FOR HYDROPONIC AGRICULTURE

A Dissertation
Presented to
The Academic Faculty

By

Sushmita Warrier

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
College of Engineering
Electrical and Computer Engineering

Georgia Institute of Technology

August 2021

**4D RECONSTRUCTION FOR HYDROPONIC AGRICULTURE**

Thesis committee:

Dr. Frank Dellaert
College of Computing
*Georgia Institute of Technology*

Dr. Anthony Yezzi
Department of Electrical and Computer Engineering
*Georgia Institute of Technology*

Dr. Cedric Pradalier
School of Interactive Computing
*Georgia Institute of Technology*

Dr. Patricio Vela
Department of Electrical and Computer Engineering
*Georgia Institute of Technology*

Date approved: 06 May 2021

## ACKNOWLEDGMENTS

I would like to thank the members of my thesis committee for their help in preparation of this work – Frank Dellaert and Cedric Pradalier, my co-advisors, who have patiently explained concepts and provided invaluable advice and guidance, but more importantly, taught me how to approach and solve problems; and Patricio Vela and Anthony Yezzi, for their critical feedback and suggestions.

Special thanks are due to the friends and colleagues who made this work possible. Antoine Richard and Amit Raj have provided significant help, resources, troubleshooting advice and their precious time in helping understand, implement and debug the volume rendering part of this work. To everyone at Borglab, I express gratitude for their unwavering support and assistance with helping me learn about Structure-from-Motion and good programming practices, in general. The people on the Hydroponics project, I thank for their support in the robot arm setup, configuration and data collection process.

I would also like to thank the Environmental Engineering team (especially Abigail Cohen) at Daniel Lab, headed by Dr. Yongsheng Chen, for the umbrella project that aims to make the world a better place and indirectly makes my thesis possible.

And last but not the least, my undying gratitude to Stack Overflow and the myriad other websites that make accessing resources and explanations so much simpler.

## CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## SUMMARY

Current plant monitoring practices are quite destructive and wasteful, requiring harvest to measure basic attributes such as weight, nutrient concentrate, etc. With this work, we attempt to develop and evaluate a precise model of the crop, allowing attribute measurement and visual monitoring, all in a non-destructive manner.

# CHAPTER 1
# INTRODUCTION

## 1.1 Overview

Hydroponic robotics introduces an efficient and accurate method for automating and monitoring the produce for quality consistency across farms. Traditional agricultural methods are facing challenges in increasing productivity to keep pace with the growing population, while dealing with a decrease in available farming resources such as water, fuel, and arable land [1], [2]. On the other hand, closed-field agriculture is growing due to advances in precision technology, sensor networks and data analysis [3]. These Controlled Environment Agriculture (CEA) systems utilize optimum growth conditions and controlled parameters for higher crop yields with increased predictability and lower costs. Hydroponic systems, such as the one described in [4] have been in research extensively to develop production systems for various high-quality, fresh crops. Another factor behind the exponential growth of hydroponics systems over traditional farming is the ability to have sustainable inventory transport and management models, with the current models and technologies expected to fail in meeting the food demands in the future [2]. However, even closed-field farming requires tools and methods to monitor and predict plant growth with minimal human intervention.

## 1.2 Research Goals

This research is part of a larger project by the Environmental Engineering Department at Georgia Tech, led by Dr. Yongsheng Chen, to study the possibility of hydroponic plant growth using treated sewage water. Modeling the plant nutrient uptake requires the inclusion of biochemical factors within the growth medium, and bioaccumulation [5]. Empir-

ical formulas such as the Michaelis-Menten equation, or ones developed in [6], describe the plant uptake and growth given nutrient uptake, modeling the relation between nutrient concentration and growth rate. While biomass is an important indicator of nutrient uptake, other phenotypes such as leaf color, age, size of plant are also crucial indicators of plant health. The study led by Dr. Chen will focus on methods to optimize growth conditions to yield desired outputs, in terms of optimizing flavor, size, nutrition content, etc.

Our objective is to model the growth of lettuce crops using a robot arm with an attached camera in a hydroponic setup, in order to inform decisions and strategies on effective plant growth. The volume estimated from the growth model will be used to predict the mass of the plant; an indicator of the nutrient uptake. We augment this with temporal association to provide the state of a plant at any point in time, enabling monitoring and studying growth pattern for different nutrient solutions. In order to find custom reconstruction strategies informed by targets for plant outputs, we evaluate three different approaches towards plant reconstruction and volume estimation. A long term goal for the project envisions the development of an automatic harvesting system, including automating the decision platform used to evaluate the readiness of the plant for harvesting.

Research into noninvasive techniques for plant phenotyping is a relatively nascent field; however, multiple approaches exist to analyze plant parameters without extensive human effort. In this thesis, we review available research and its applicability to our use case, and propose and evaluate three different approaches to model plant growth and estimate volume.

The thesis is arranged as follows: We describe the hydroponic setup we use in Chapter 2. The subsequent three chapters are self-contained modules of each of the three approaches we used during the thesis to estimate the volume, and the final chapter discusses the results, tradeoffs and evaluates the accuracy of the approaches used, along with a discussion of best practices for capturing images for each approach.

# CHAPTER 2

# EXPERIMENTAL SETUP

## 2.1 Growth Chamber setup

This section details the setup of the growth area and the process used to collect and store the imaging data. The hydroponic growth chamber used in our research is located in the Daniel Lab at the Georgia Institute of Technology, Atlanta campus. It is a roughly 2.43x2.43x2.43 $m^3$ chamber, and there are 6 "grow towers" on each side of the room. The temperature is maintained at approx 24° C and 55% RH. Each grow tower was initially planted with 8 saplings, which were then periodically harvested for mass measurement. Each side of the room is lit by around 6 tubelights, arranged in 3 equally spaced rows. The distance between the lamps and the towers was around 38.1$cm$ initially, with the distance progressively decreasing as the plants grew. The lighting structure was a soft limit for the movement of the robot in the Z-direction(considering Z to be the direction moving directly towards or away from the plant). While the lights could be temporarily moved further back to accommodate the robot, it was not very easy to move around and hence was moved only when required. This setup is shown in Fig Figure 2.1.

Figure 2.1: Hydroponic Chamber setup

The 4 DOF robot arm is mounted on a base connected to a vertical struct wherein it can

be slid upwards and downwards, as shown in Figure 2.2. The vertical metal struct on which the arm has been placed is in fact a placeholder for a cable robot under development. The movement of the robot base in the vertical and horizontal directions was done manually, with any horizontal movement requiring the entire struct to be moved to the desired position. A Raspberry Pi v2 camera with a Sony IMX219 8-megapixel sensor capable of HD imaging is attached to the end-effector of the robot arm.

Figure 2.2: Robotic arm mounted on a struct

## 2.2 Data Collection Process

The robot is placed between the lamp and the plant, maintaining a distance of around 5-10 cms from the plant, in order to get the leaves in focus. The trajectory followed was that of concentric planar circular motion, with the number of circles increasing with the growth of the plant. The center of the image was set towards the base of the plant. The initial growth stages required only one traversal of this trajectory to capture the entire plant, while the final growth stages needed around 4 circles to completely capture the plant. The issue with this trajectory was that the leaves at the bottom of the plant did not grow into a flat structure, but were near perpendicular to the plant due to heliotropism. This resulted in multiple frames from the image capture being occluded by the underside of these leaves during the circular motion of the arm. To correct this, we then modified the trajectory to follow a more elliptical, non-planar trajectory, with the center of the image corresponding to the

plant height; which led to the camera taking a higher position relative to its initial position when imaging the bottom leaves of the plant. This results in a relative improvement in the data collection, with far fewer frames having obstructing views (Figure 2.3). The number

Figure 2.3: Obstructive views

of images taken per circle also varies with the plant growth. Some samples from the dataset collected are shown in Figure 2.4. These images are samples of the data collected from the same plant over a period of 7 weeks, and are not a complete representation of the plant.

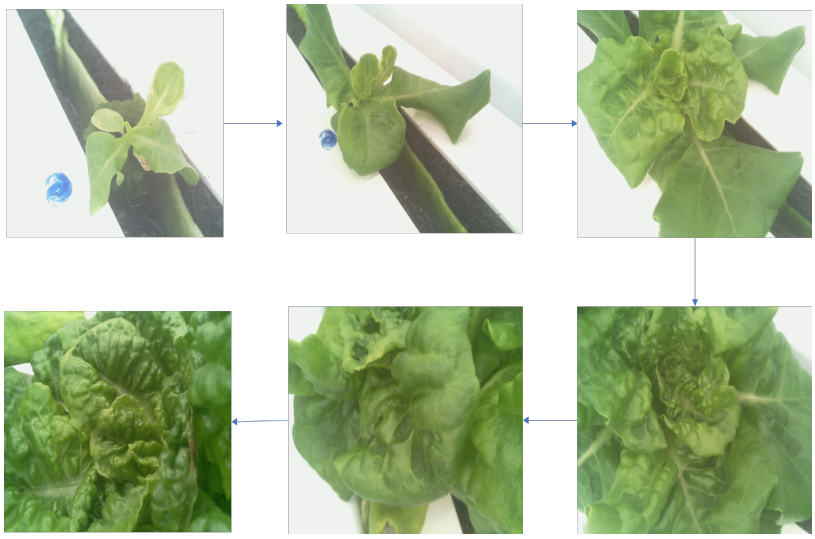


Figure 2.4: Initial dataset samples over 7 weeks

In all, we captured 10 lettuce heads over nearly their entire growth stage- from sapling to full growth. Images were taken thrice a week. It is to be noted that the plants suffered a dehydration shock around week 8, leading to many of them drooping and not having a

proper revival before the final harvest. Attempts were made to capture the plants scheduled for harvest thrice a week so as to have data points correlating the reconstruction with measured weight, but due to scheduling conflicts with the harvesting team it was not always possible. Only 17 out of the initially sown 96 plants were measured before their harvest; of which 9 were healthy, and 8 plants were captured after some revival from the dehydration shock. The entire dataset is stored on a Dropbox Professional folder, and can be viewed at https://www.dropbox.com/sh/r2hu8pmaytcfebb/AAAbCEdvWssWr31smd644DHKa?dl=0.

# CHAPTER 3

# PLANT AREA BASED SEGMENTATION

## 3.1 Introduction

In this chapter, we detail a color-based segmentation used to compute the leaf area. We posit a proportional relationship between leaf area and its weight, and describe the approach used to segment and compute the leaf area. We also test our hypothesis by predicting wet weight of a crop (weight including water content) based on the computed leaf area.

## 3.2 Related work

From an agricultural perspective, research on non-invasive phenotyping methods has branched into various streams - from leaf area correlator to volumetric imaging [7]. The Controlled Environment Agriculture (CEA) Program at Cornell University [4] uses computer models to simulate hydroponic lettuce production under variegated environmental conditions, the results of which were used to develop a pilot greenhouse facility at a commercial scale.

[7] reviews noninvasive techniques used for phenotyping plant parts from lab-to-field (sensors being taken to the field), such as RGB Imaging, Chlorophyll Fluorescence, Thermal Imaging, and Imaging Spectroscopy. Of particular interest is the success in finding highly significant linear or polynomial correlation between calibration of plant part area based on total leaf area and fresh and dry mass, via RGB imaging. An increase in precision can be achieved by digitally reconstructing leaf area and growth rates. While this paper focuses on plant shoots, there seems to be potential to generalize this correlation for other plants using simply leaf area as well. Other sensor based methods such as those reviewed in [8], or coupled with Bayesian Networks [9], have been found to be highly dependent on environmental parameters, leading to imprecise yield predictions for variable climatic

effects [3]. [10] evaluates the effectiveness of different colorspaces in crop segmentation from background, while [11] uses color-based image processing methods to segment lettuce in a hydroponic setup. [12], on the other hand, uses marker-based detection to compute scale-invariant lettuce leaf area. We favor modeling the plant using RGB imaging over the other methods such as multispectral imaging due to the low cost of the cameras and ease of availability.

## 3.3 Approach

### 3.3.1 Leaf Area Index

The Leaf Area Index (LAI) is a dimensionless measure to quantify plant canopies. For broadleaf canopies, it is defined as the ratio of one-sided leaf area to unit ground surface area [13]. LAI depends on plant density, light sources, and leaf orientation, in addition to canopy size, and is used as an indicator of plant growth [14]. The amount of light received by the plants (directly proportional to the leaf area) is correlated to the photosynthetic production and evapotranspiration; which can be used to predict biomass, an important indicator of nutrient uptake. An inverse exponential relation exists between LAI and light received [15], [16], expressed as:

$$P = P_{max}(1 - e^{-c.LAI})$$

, where $P_{max}$ is the primary production, and c is a coefficient specific to the crop being modeled.

LAI can be measured in multiple ways [17]:

1. Direct Methods: After harvesting, the leaf area can be measured in one of three ways.
    - Manual measurement: As the name suggests, leaves are manually measured.
    - Planimetric measurement: This method uses a plant planimeter to measure leaf

perimeter, derives plant area based on the measured perimeter, and computes their ratio to the ground.

- Gravimetric measurement: This method uses the dry weight of the leaves to estimate biomass, which is then used to compute leaf area [18].

Due to individual leaves being measured, direct methods are the most accurate way of measuring LAI; however, they are also highly inefficient.

2. Indirect Methods: Indirect methods are gaining popularity due to their efficiency, speed and non-destructive nature.

    - Inclined Point Quadrats: Primarily designed to measure LAI in short canopies, it expressed measured foliage as a percentage of the ground area which bounds the quadrats.
    - Digital Canopy Analysis: This approach uses image processing-based methods to compute LAI from canopy undersides. Of all the methods, this is the fastest and most accurate approach to computing LAI.

The aim here is to provide an approximate estimate of mass, based on leaf area. Biochemicals under analysis are expressed as a ratio of weight of biochemical to the total weight, including water present, of the lettuce.

### 3.3.2 Segmentation

Here we attempt to compute the leaf area index by computing the leaf area in images. Due to the pandemic, the hydroponics setup was moved to researchers' homes as best as possible. Images of 35 lettuces with measured wet weight (weight before the lettuce is dehydrated) were taken from a handheld camera, with a cluttered background. These lettuce were divided into 4 reservoirs, but for our purposes, we consider them part of the same dataset. Front facing and top down images were captured; however, not all lettuces

had both kinds of images, due to which we used only the front facing images, of which samples are shown in fig Figure 3.1. A ruler was placed next to the lettuce for scaling to physical units.

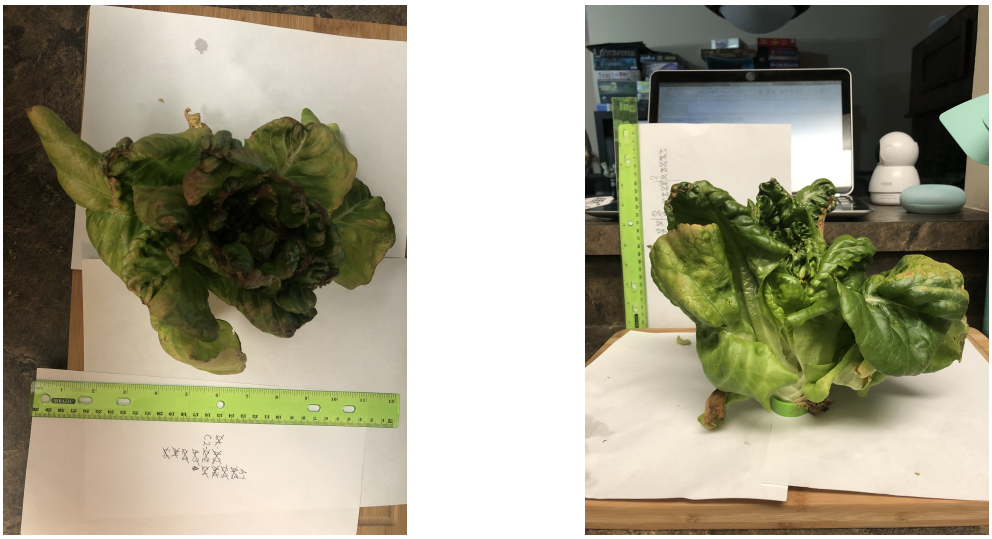


Figure 3.1: Samples of lettuce images taken without robot setup

As a first step, we segmented out the lettuce from the background. Due to the noisy background with various green-colored objects close to the lettuce, and high lighting variability, simple color-based segmentation failed to accurately and reliably capture the green areas of the leaf, as shown in Figure 3.2. Even the relatively well segmented plants left much to be desired in accuracy. To this end, given the limited number of images, we chose to use a commercial photo cropping tool [19] to segment the lettuce with manual input. We then computed the area of the lettuce in pixels, and rescale the area to cms using scale information extracted manually from the images.

The relation between measured weight and plant area is approximated to be roughly linear, and hence we feed the data to a linear regression. Reserving 20% for validation, we fit a linear regression to the training data to predict wet weights from the leaf area in $cm^2$.

As seen in Figure 3.4, we observe a high degree of precision in the validation set, with an R2 score of 0.94.

Since the lettuces could not be dehydrated, we assume a proportional relationship be-

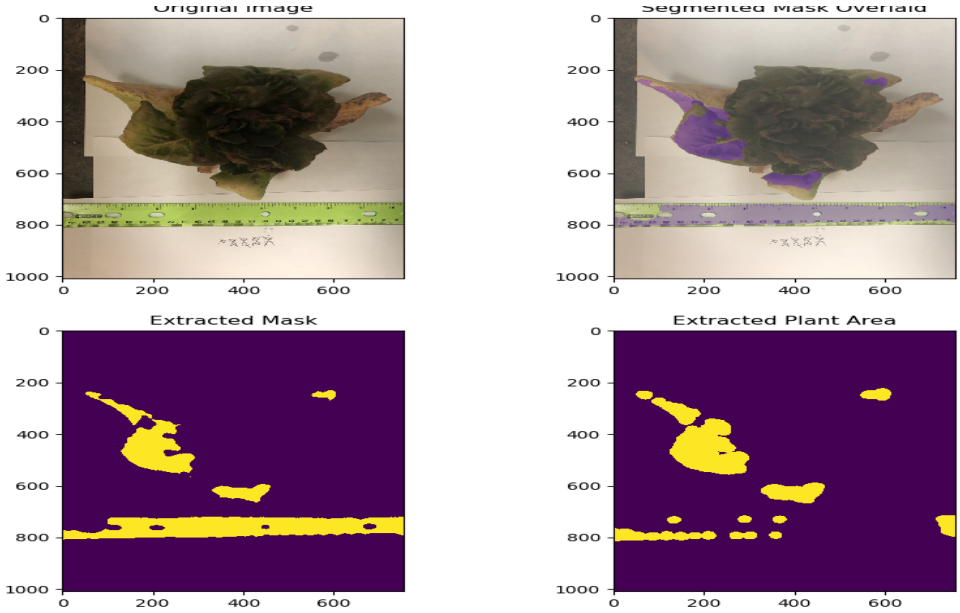



Figure 3.2: Segmentation failures in lettuce images

tween wet weight and nutrient uptake, and extrapolate the rough prediction of lettuce wet weight to prediction of dry weight, given adequate and accurate training data; thus allowing precise measurement of biochemical composition of the lettuce.

## 3.4 Summary

Here, we described the challenges faced in cleanly segmenting the lettuce from a cluttered background, and the subsequent approach taken to estimate weight from leaf area. While this approach cannot be used for volume estimation, it provides a good approximation for plant growth quantification.

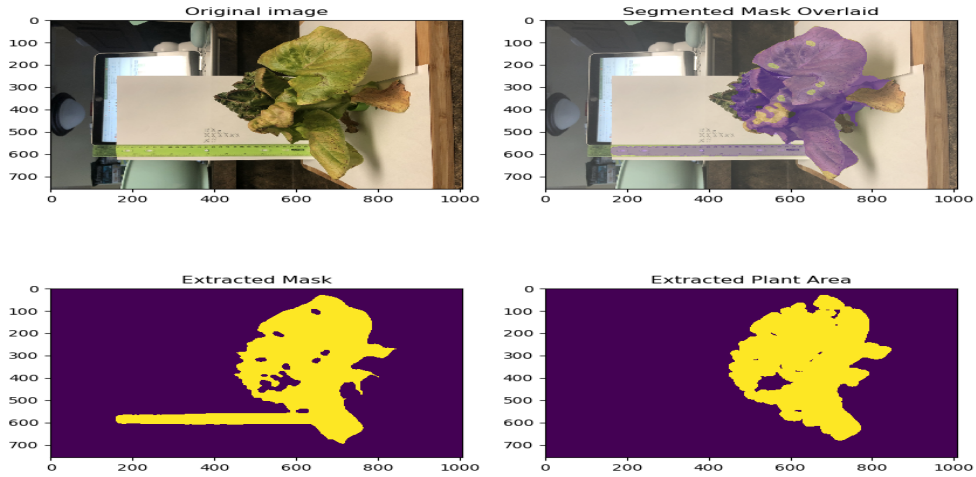
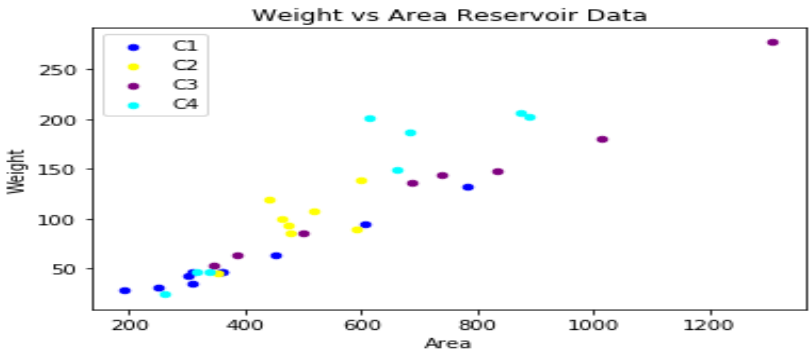
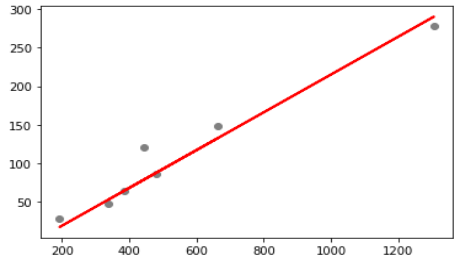



Figure 3.3: Relatively well-segmented lettuce images




(a) Weight vs area graph for training set



```
[10]: # Metrics for model performance
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_test, Y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(Y_test, Y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y_test, Y_pred)))
print('Root Mean Squared Log Error:', np.sqrt(mean_squared_log_error( Y_test, Y_pred )))
print('R^2 score:',metrics.r2_score(Y_test, Y_pred))

Mean Absolute Error: 12.867356054786006
Mean Squared Error: 335.627924076105
Root Mean Squared Error: 18.320150765648872
Root Mean Squared Log Error: 0.25232660841334303
R^2 score: 0.9455694306517867
```

(b) Regression results on validation set

Figure 3.4: Raw data and linear regression on lettuce area

# CHAPTER 4
# CURVE BASED RECONSTRUCTION

## 4.1 Introduction

In this chapter, we propose a curve based reconstruction approach to create a 3D model of the lettuce. We describe the algorithm used, and propose contributions to the framework that we hope will increase reconstruction quality.

## 4.2 Related work

[20] compared reconstruction from digital photographs and terrestrial LiDAR, and found LiDAR better suited for their analysis. Other methods investigated include 3D Time-of-Flight Camera [21], LiDAR [22], X-Ray Tomography[[23], [24]], and Magnetic Resonance Imaging [[25], [26]].

Papers such as [22] discuss use of stereo-imaging for 3D reconstruction of plant canopies, while [20] extracts color characteristics of plant canopies using 3D reconstruction. [21], which uses stereo and Time-of-Flight cameras to create 3D reconstructions of individual leaves, seems to have a similar hydroponic setup as ours, but their data collection process slightly varies. However, all these approaches either use multiple commercial scanners, or integrate interactive components to the 3D reconstruction process, making them unsuitable for our use.

The challenges of modeling a lettuce via vision based methods include repeated texture on the leaves, the sparsity of usable features during the earlier stages of plant growth and the high degree of occlusion present in the plant. Given precise calibration and geometric information available to us due to the robot arm, we attempt to use Structure from Motion to construct a point based 3D reconstruction of the lettuce. Related work[28] uses

a SLAM based pipeline with a point based feature matching front end to reconstruct a sparse reconstruction of field crops with multisensory input, including GPS and IMU. [25] uses a multiview approach to jointly optimize keypoints over multiple images, while [18] segments images to extract ROIs from object contours, which are then used to generate features. However, this approach faces 2 problems with our dataset: 1. A significant number of features are detected at the leaf boundaries, making for unstable matches, and 2. In the early growth stages of the lettuce, there isn't enough detail on the leaf to extract features. The [http://trimbot2020.webhosting.rug.nl/——Trimbot 2020 project], a European Union Horizon 2020 programme conceived to create an autonomous garden bush trimming robot, uses optical flow and disparity maps to reconstruct 3D point clouds based on stereo pairs, with attached 3D motion vector. Yet other papers rely on Deep Learning for plant classification with handcrafted features[29] or a combination of multiview SFM with commercial software to generate dense reconstructions[30]. [19] evaluates some popular interphotogrammetry software, some of which we review in Sec [sec:Software-Review].

Direct Sparse Odometry [27] is a visual odometry formulation combining a direct probabilistic model with a joint optimization of model parameters. We attempted to run some of our earlier collected data on the DSO software, but were met with poor results for the reconstruction, which is also reviewed in Chapter 6.

Curve reconstruction, on the other hand, uses edge information, which is more easily available(and a better representation of the object), to reconstruct curves in an SFM pipeline. [35] uses a Rao-Blackwellized fitting of piecewise smooth subdivision curves to reconstruct objects. It however requires the object to be cleanly segmented from the background, which is slightly difficult to achieve in our setup, as discussed in Sec [sec:Experimental-Setup-and]. [34] uses a point based curve reconstruction method, representing 2D curves as a sequence of 2D points, along with a reference curve to minimize the curve reprojection error. [33] uses factorization for the reconstruction of general curves, while [36] generates an ordered set of sampling points and minimizes an energy function to construct 3D curves

with projection segments that self-overlap.

Of particular interest are papers that aim to augment point-based reconstruction with curve information- [32] reconstructs general curve shape and camera pose from multiple views, even with partially-visible curves in each view; thus optimizing points, curves and poses simultaneously. [26] proposes a framework for 3D curve reconstruction, generating as output an unorganized set of curve segments, meant to serve as a base structure for surface reconstruction. This paper assumes coarsely calibrated cameras, and hence is divided into two sections (one producing a coarse 3D sketch, and the subsequent stage refining camera poses to generate a detailed final sketch); however, this can be simplified given our availability of precise camera poses. This approach produces fragmented, unorganized curve sketches. Usumezbas [24] builds on the work in [26] by integrating topological and structural information about the 3D curves, resulting in a more semantically coherent representation. This can be specially useful to us when modeling the growth of overlapping leaves in the lettuce head.

## 4.3 Approach

### 4.3.1 Structure-from-motion: An Introduction

Structure from motion (SfM) is a photogrammetric method to estimate 3D structures from multiple overlapping 2D images. The problem statement here is to find the correspondences between 2D images and their 3D reconstruction, along with recovering the camera parameters (pose and calibration). Figure 4.1 shows the stages in a global SfM pipeline:

Below, a brief overview of each of the stages is provided:

1. Feature Extraction: 2D features from images are extracted via various algorithms such as SIFT/ORB/AKAZE, etc. [27], [28], [29]. The features are described using a numerical descriptor, such that they are scale and rotation invariant.

2. Feature Matching: Features corresponding to similar descriptors in images are matched

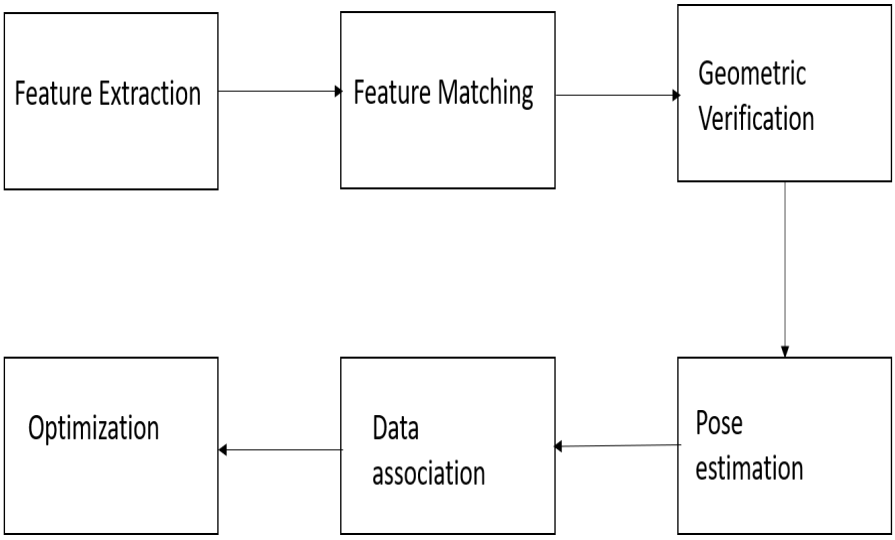



Figure 4.1: Global SFM pipeline

as possibly being the same feature in a different image, resulting in putative matches.

3. Geometric Verification and Pose Estimation: The putative matches are verified on the basis of the fundamental/essential matrix computed from the set of matches. In a typical implementation, RANSAC is used to estimate the best possible F/E matrix, by sampling sets of points and choosing the computed E matrix with the most inliers. This E matrix is then used to recover relative pose between pairwise matches.

4. Averaging: Rotation and translation averaging methods are used to estimate good initial global poses by estimating a rotation matrix/translation vector on each graph vertex that best preserves the relative poses on connected edges [30].

5. Triangulation and Bundle Adjustment: Data association involves creation of feature tracks in images, and using triangulation [31] to initialize estimates of 3D points corresponding to the projected features in the images. This is then fed to bundle adjustment, which optimizes camera parameters and 3D points simultaneously using algorithms such as Levenberg-Marquadt.

The output of SfM is a sparse point cloud representation of the 3D structure. While further dense reconstruction methods are also available, they are out of the scope of this thesis.

### 4.3.2 Challenges with point based reconstruction

Classical SfM techniques do not work very well with the lettuce structure for a few reasons:

- Feature points often detected in leaf folds, which are not always visible across multiple views, leading to poor matching.
- DDMV (GTSFM front-end) showed reconstruction reliably possible close to base of lettuce; the ends of the leaves were very messy when reconstructed.
- Due to high texture repetition of lettuce, matching could not be reliably done over all images.

In addition to repeated structure in the lettuce, the relatively featureless leaf surfaces during the early stages of the lettuce also pose a challenge to SfM. Figure 4.2 shows SIFT keypoints detected on one of the images. As can be seen, quite a few keypoints are on the folds around the leaves, and in other distinct background objects.

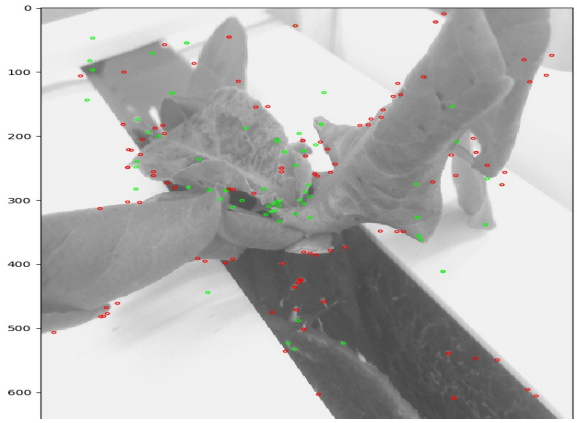



Figure 4.2: Keypoints detected on lettuce

### 4.3.3 Curve based reconstruction

As discussed, point based SFM does not lend itself well to repeated texture. To counter this, we look at contour based matching. Leaf contours, ie. curves, are a better approximator of

lettuce shape than random interest points, due to their high ability to be distinguished. In addition, the entire curve does not need to be visible at all times when attempting to match the images, making it robust to viewpoint changes.

*Edge Detection*

Among the many edge detection techniques available, Canny might be the most popular method. The 5 step process uses a double thresholding on a Gaussian-filtered image to detect potential edges, and applies hysteresis to filter out weak edges. Running Canny on fully-grown lettuce resulted in the edge detection as showed in Figure 4.3. The veins in the lettuce, along with the overlap make clean detections very difficult, which could potentially cause issues with the shape descriptor.

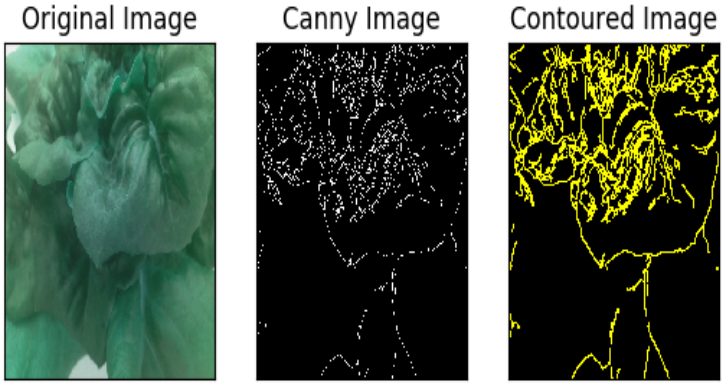



Figure 4.3: Canny Detection on Lettuce

To extract the edges of a lettuce, we use a deep learning-based edge detector called Holistically Nested Edge Detector (HED) [32], which creates nested holistic layers of edges, with the highest layer being the outermost leaf contours, and successive layers adding details such as veins in the leaf.

HED aims to address multilevel feature learning for holistic edge detection by end-to-end image prediction. It uses fully convolutional neural networks to output an edge map as a classification problem, and deeply supervised networks, which guide early classification

results. Each layer of the neural net also has an associated side output, which is passed through a classifier, and is used to guide the outputs towards the desired prediction. This is done by implementing a weighted fusion layer which combines multiscale outputs and learns the combined weights. These side-output edge maps are integrated to form a unified output map using a standard stochastic gradient descent optimization.

The HED architecture automatically learns detailed hierarchical features, with progressively fine-tuned edge maps that outperform other contemporary methods [32].

As can be seen in Figure 4.4, we compute the edge maps of the lettuce using HED, resulting in an outer-level curve-representation of the lettuce leaves. For the same lettuce input that we provided Canny, we observe cleaner, more coherent edge detections that better represent the lettuce shape. We apply non-maximal suppression (NMS) to the HED

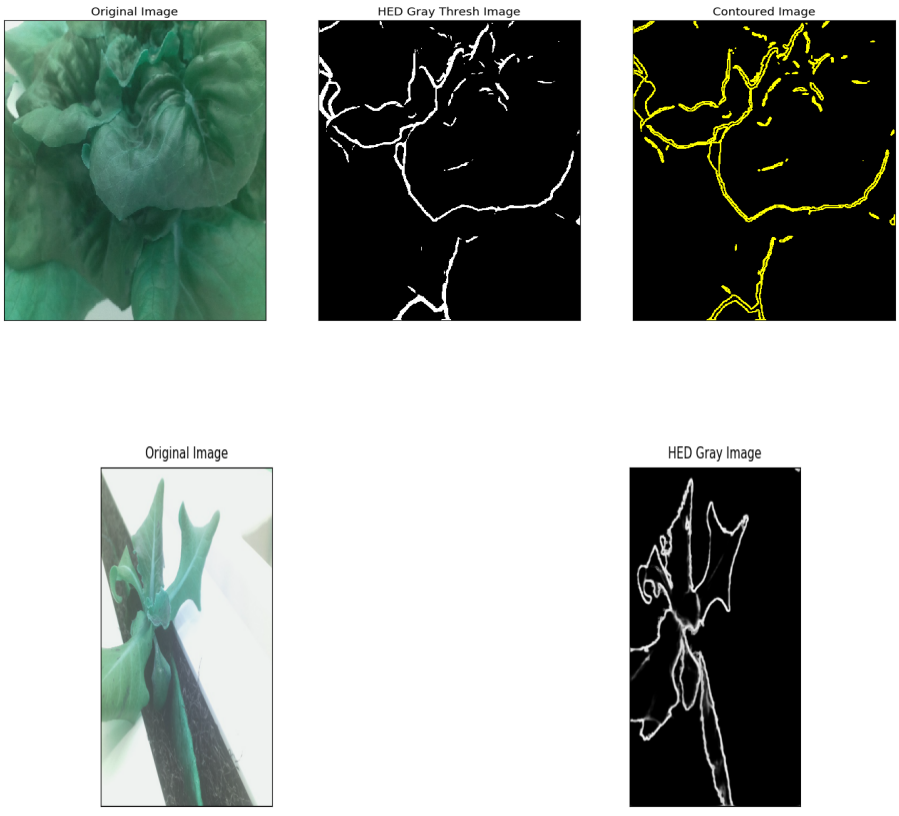



Figure 4.4: HED Edge Detection

outputs to suppress the edge thickness, and use these contours as input to the next stage of the feature descriptor process, which we expect to suffice for a broad shape reconstruction of the lettuce. Additional details in the form of leaf veins can be added for a better reconstruction and visualization of phenotypes.

*Shape Context Descriptor and Matching*

Shape Context was introduced as a sparse, but robust shape descriptor in [33] and further demonstrated in shape matching algorithms in [34], [35]. The underlying assumption that this approach shares with curve reconstruction is that the shape (or curve in our instance) is described as a set of points sampled from the contours of the object, such as from an edge image. For pairwise images, SCD attempts to find the best possible shape similarity as a correspondence problem, similar to feature matching in SfM.

For each point *p* on the contour of the object, we consider the set of vectors from *p* to all other points on the contour, which would be of size n-1, where n is the number of points on the contour. This set encodes the position, and hence shape, of all points on the contour relative to each point, forming a rich descriptor. However, for a large number of points, this becomes a bulky descriptor describing the shape in too much detail, leading to 'overfitting' when attempting to match points, causing failures on different viewpoint changes. Instead, we use the distribution over positions of pixels as a robust and sparse descriptor. Therefore, for point *p*, we use a coarse histogram of the remaining coordinates over uniform log-polar bin space, such that

$$h_i(k) = \{q! = p_i; (q - p_i) \in bin(k)\}$$

where q are the remaining points on the contour, is the shape context descriptor for *p*. Using the log polar space makes the SCD sensitive to nearby pixel changes, effectively encoding neighboring-points information as would be observed in a curve segment.

The SCD algorithm further computes a matching cost using a chi squared statistic,

represented by

$$C_{ij} = \frac{1}{2}\sum_{k=1}^{K}\frac{(h_i(k) - h_j(k))^2}{h_i + h_j(k)}$$

for K-bin histograms at points $p_i$ and $q_j$ on the shapes to be matched, respectively. The algorithm treats the cost matrix containing costs for all points on the shapes to be matched, as a weighted bipartite matching problem, with the objective to minimize the total one-on-one matching cost. The Hungarian algorithm [36], an algorithm designed to solve the assignment problem, is used to minimize the cost matrix and find optimal matches.

SCD is scale and translation invariant, and while there exist modifications to add rotation invariance, we do not believe it necessary for our particular use case.

In order to match different images of a lettuce, we detect points on the contour of the plant. The resultant histogram from the SCD is descriptive enough to represent the shape of the lettuce as a function of the distances between points on the contour.

The point-to-point matching correspondences after the Hungarian algorithm were not accurate, as seen in Figure 4.5. We posit that the results are due to the following two

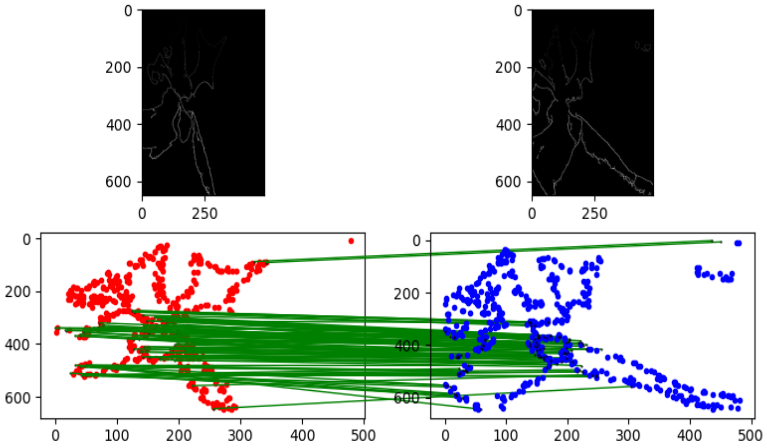



Figure 4.5: Verified matches

factors:

- high motion between images ( 30deg) when images were captured by the robotic

arm, making observation of features in subsequent images difficult, and

- high symmetry between leaves, making certain sections of the leaves similar enough for the neighborhood encoding to not be effective.

Here, we have not utilized the full information from the robot to estimate good matches. In order to improve results, we separate the reconstruction into a 2 step approach:

- Camera recovery
- Structure recovery

*GTSFM*

GTSFM [37] is an end-to-end global SfM pipeline, designed to utilize Dask to perform parallel computation. It is heavily based on GTSAM, a library implementing sensor fusion in robotics using factor graphs.

The basic GTSFM pipeline is described in Figure 4.6

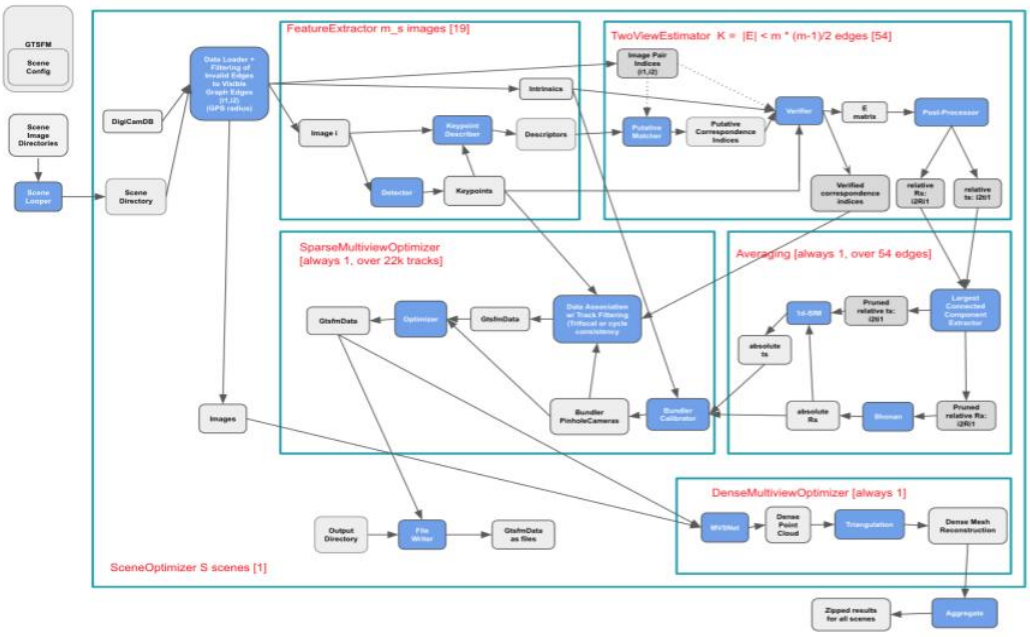



Figure 4.6: GTSFM pipeline

The DDMV front-loader evaluates combinations of available modules for the different stages of the front-end (feature extractor and descriptor, matching, and verifier) to find the optimum method for each of these stages based on a common set of metrics. This allows GTSFM to evaluate various new deep-learning based methods for front-end, as well as incorporate parallelization across datasets for two-view estimators, which determines pairwise matches and computes the essential matrix. The relative poses computed from E-matrix are then fed to the averaging modules, Shonan rotation averaging [38] and translation averaging (which includes outlier removal and 1dSFM [39]), that return optimal global rotations and translations. The averaging methods are crucial to global SfM, for they make the pipeline robust to outliers. These are combined to a Bundler calibration, and are fed along with the verified matches, to the data association module.

The data association module aims to find 2D to 3D associations. It does so by creating feature tracks of keypoints in images, which are then triangulated to form an initial estimate of the 3D point corresponding to the observed feature. Of the available modes for triangulation (simple- which considers all points as inliers, and RANSAC [40]), we use RANSAC-based triangulation for a more robust estimate. We then filter out bad estimates by removing all measurements with a reprojection error greater than a user-defined threshold.

The last step of a typical SfM module is Bundle Adjustment. We use the tracks and initial cameras and 3D point estimates and optimize it by minimizing the reprojection error of predicted and observed image points. GTSAM offers a variety of optimization classes, of which GTSFM uses the Levenberg-Marquadt algorithm, implemented with a nonlinear factor graph, as an optimizer. As an output, this module returns optimized intrinsics, optimal poses, and a sparse 3D point cloud; of which we are interested in the poses as a way to improve matching in the SCD algorithm. In the vanilla pipeline, however, we notice that we do not use the robot information available to us in order to improve putative matches.

Given the use of a robotic arm to capture images around lettuce, we have rough camera

locations, and therefore, strong priors on the poses. Using this information, we can improve the estimates of the cameras recovered after bundle adjustment. These precise cameras can then be used to improve the putative matches in the dataset, leading to better reconstruction.

In addition, the verified matches can be optimized in the epipolar search stage due to the controlled data capture setup, by integrating known depth of the object (lettuce) from the camera.

Range limited epipolar check: When a 3D object is viewed from distinct positions, the 2D image is formed by perspective projection. This is modeled as rays passing through the focal center of the cameras(which are modeled as a plane), intersecting in a point in space, which is where the corresponding point of the 3D object is located. [41] Since the camera

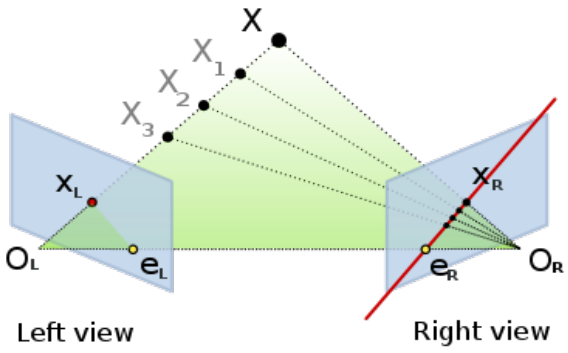



Figure 4.7: Epipolar geometry

centers are distinct, each point in one camera projects to a distinct point in the second camera. These points, called epipoles, denoted by $e_1$ and $e_2$ are used to provide constraints in the feature matching process. In a regular matching pipeline, the epipolar search area is the points on the epipolar line, which spans the entire length of the image. However, given the knowledge of the distance of the object from the camera plane(ie. depth), the projection of the 3D point in the second camera, would be within the segment which has endpoints defined by origin of camera plane to the known depth. The following pseudocode describes the algorithm:

***Range Limited Epipolar Check Pseudocode:***

For a given pair of images, we have the strong priors on camera calibrations: the poses of the cameras and the intrinsics. For two points $p_i$ and $q_j$ on images i and j, respectively:

1. We create a direction ***D*** of $p_i$ in camera frame i.

2. We use the min and max depths to create two corresponding points in camera frame i, and subsequently convert them to world frame.

3. The end points of the line segment are the pixel coordinates of the projection of the above points in the image frame j.

Given the minimum and maximum depth, the epipolar search segment is now defined by the endpoints corresponding to the min and max depth, thus improving robustness and accuracy during the verification stage. Here, the putative match(ie. the feature point in image 2 corresponding to the same feature point in image 1) is verified by measuring the distance between the point and the epipolar segment it is expected to be in. The following pseudocode describe the computation, based on [42]:

***Pesudocode for point to segment distance computation:***

For start and end points $X_1(x1, y1)$ and $X_2(x2, y2)$ of the epipolar segment, and a matched feature point $q$ in image j:

1. Convert line segment and point to vectors. Vector representation of the point is relative to the starting point of the line segment.

2. Convert line segment $l$ to unit vector.

3. Consider the line extending the segment, parameterized as X1 + t (X2 - X1). $t$ is the parametrization of the segment; ie. how far along the line segment the projected point falls. We want to find $t$ that minimizes the distance from point to segment.

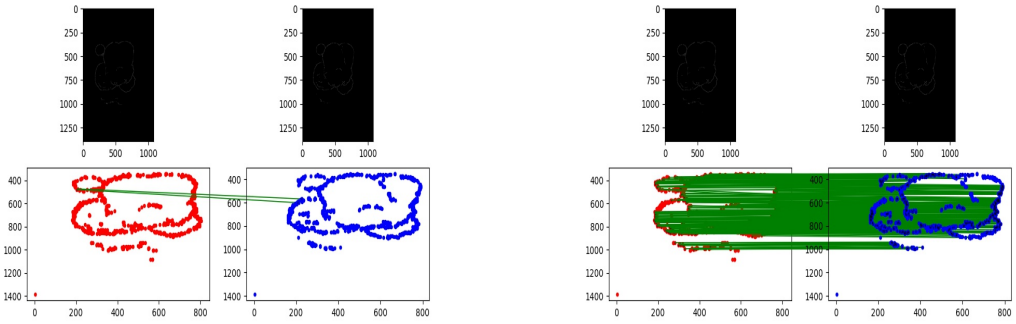
4. Get $t$ between 0 and 1. If $t < 0.0$ or $t > 1.0$, set $t = 0.0|1.0$ .In this case, the closest point to $q$ is one of the segment's end points(t=0 means the projection is right

at (x1,y1), and t=1 means projection is at (x2,y2)). If $0.0 < t < 1.0$, the closest point lies on the segment.

5. Get projection on segment, using $p_{proj} * t$. Scaling line vector by $t$ gives point on segment corresponding to matched point. Calculate distance between the projected point $p_{proj}$ and $q$. To get actual nearest point, add X1 to $p_{proj}$.

For a distance smaller than a certain threshold, we accept the match as verified.

As the lettuce grows at an exponential rate, and the camera was moved to accommodate the increase in size, the min and max depths are empirically chosen based on dataset properties. Below, we show matches in pairwise images with and without integrating the pose information in estimating the putative matches.



(a) Shape matching without pose information (b) Shape matching with pose information

Figure 4.8: Pose prior integration comparison

However, we also find pairs of images without matched points, as shown in fig Figure 4.9.

It could be argued that the representation was not dense enough to encode adequate neighbor information, or that the shape was not descriptive enough; but we did not see an improvement despite increasing the number of sample points by 2x.

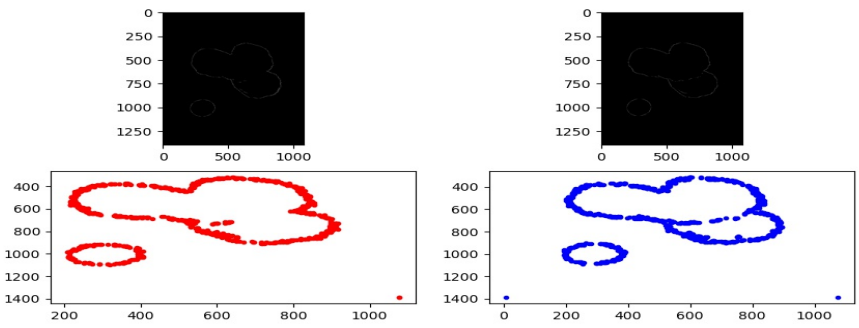



Figure 4.9: Shape matching failure even with pose information

*Backend*

The backend of an SFM pipeline is relatively straightforward. Once we have the verified matches, we build feature tracks using disjoint set forests to create tracks of the same feature across multiple images. We then triangulate the features from the tracks to create an initial estimate of its corresponding 3D point. For GTSFM, as mentioned, we have multiple modes of triangulation available- with the default being RANSAC based, due to its robustness to outliers. The final step of this process is optimization, or Bundle Adjustment. As mentioned earlier, GTSFM uses a global BA pipeline, which means that all cameras and 3D points are resolved simultaneously via a non-linear least squares minimization algorithm, such as Levenberg-Marquadt.

GTSAM [43] is a library based on factor graphs [44], designed to implement optimization frameworks for SLAM and SFM. Factor graphs are probabilistic bipartite graphical models that are used to model problems in estimation. The main components of a factor graph are factors and variables. Factors are probabilistic information on variables, based on measurements or priors, and are connected to variables. Variables are the unknown variables in the estimation problem. GTSAM offers various easily-implementable functions to create factor graphs, add measurements, priors, and initial estimates, and optimize them. Here, the factors provide the nonlinear squared reprojection error. For a nonlinear graph such as ours, the optimizer iteratively linearizes the graph to minimize the error. We gener-

ally add priors to the first pose to mark it as origin of the world system; but given the robot pose priors available to us, we can add pose priors to every camera.

GTSAM also provides the covariance matrix post incorporating measurement information. The covariance matrix is an NxN matrix (where N is the number of poses), describing the covariance (joint variability of two variables) of all pairs of poses. For a factor graph with pose priors not added, the uncertainty risks increasing out of bounds in all dimensions, as measurement uncertainty increases. However, by adding prior pose information via unary factors, the uncertainty is constrained within bounds, providing better optimization results.

One of the possible reasons of BA failure, that we noticed on our dataset, was the lack of enough tracks detected across the images. Fewer tracks and smaller track length meant there are possibly cameras that do not correspond to a measurement, leading to an inference error during optimization. For instance, our pipeline detected only 31 3D points across 75 images, leading to multiple triangulation failures that lead to a complete failure in optimization. We believe this can be resolved with a different dataset capture, described in detail in chapter 6, and a different matching algorithm.

## 4.4 Summary

Here, we implemented curve based reconstruction for lettuce, which we deem to be advantageous due to their relative independence from view-dependent feature point detection on lettuce folds. The neighbor position encoding provides more information than a feature point, while not requiring the entire curve to be visible at all times. We also improve putative matching by integrating pose priors to perform range limited epipolar checking, and optimization. We note the challenges faced during reconstruction, and provide proposals for improvement.

# CHAPTER 5
# NEURAL VOLUME RENDERING

## 5.1 Introduction

In this chapter, we propose a neural rendering approach for lettuce reconstruction. We describe the related literature, our approach, and the challenges faced in neural network learning and rendering.

## 5.2 Related work

We investigate the use of volume rendering as a precise, detailed representation of the lettuce, providing a more direct and accurate way to estimate the volume of a lettuce. Neural rendering approaches use, as the name suggests, neural networks to sample, represent and render the images. A formal definition, as defined in the review paper [45] is ”deep image or video generation approaches that enable explicit or implicit control of scene properties such as illumination, camera parameters, pose, geometry, appearance, and semantic structure”. While most papers target novel view synthesis as their application, we propose to compute the volume based on creating a mesh of the rendered model in already available views.

From an agricultural perspective, neural network-based methods are still in its nascent stage. [46] uses Deep Learning to detect and classify plant disease, while [24] reviews the use of machine learning models in various stages of an agriculture supply chain, including crop yield prediction and harvesting. The majority of DL applications in agriculture are geared towards weed identification, plant recognition and classification [47]. Most of these approaches are specialized for outdoor agriculture, and do not, to the best of my knowledge, attempt to reconstruct the crop.

Papers such as Occupancy Networks [48], IM-Net [49], and DeepSDF [50] were among the first to use neural networks to define binary occupancy through feature vectors and 3D coordinates, or regressing signed distance functions from 3D coordinates. These models, while pioneering at the time, did not capture fine details in the rendered models. Papers such as PiFu [51] improved on learning detailed implicit representations using SDFs by reprojection into a pixel-aligned representation. Deep Local Shapes [52] stores a grid of latent codes, each containing information about a local neighborhood, reducing excessive memory requirements. While the previous papers are restricted to mesh and voxel based representations, thus suffering from low resolution, the Differential Volume Rendering paper [53] combines an implicit scene representation with a differentiable rendering approach, making learning possible directly from RGB images without storing volumetric data, and resulting in watertight meshes.

Neural Volumes [54] uses an encoder-decoder network to represent 2D images in a 3D volume representation, along with a ray marching algorithm to integrate color and opacity information. The rays are queried on a discretized RGBA voxel grid, as well as a 3D warped grid structure, reducing artifacts. However, it requires the background to be captured separately in order to render objects within a bounded volume of the background.

DeepVoxels [55], and its successor Scene Representation Networks [56], use an implicit multilayer perceptron (MLP) to represent a continuous scene through a feature vector. Here, they use the feature vector at any 3D point along the ray to determine the subsequent step size, finally decoding into a single color for each pixel. SRN often results in blurry and distorted renderings for fine details on the object, which we deem important in order to study phenotypes of plants and monitor their growth and plant health.

Another approach we investigated was neural radiance fields [57]. The algorithm synthesizes views by ray marching to generate a 5D representation of the scene, using fully-connected non-convolutional neural networks to transform them into sets of colors and opacity, and then using classical rendering methods to develop a 2D image. Representing

a scene as a Neural Radiance Field results in high quality, detailed 3D rendering of the object from multiple views; making it potentially extremely promising for modeling a lettuce head with high vein texture on its leaves.

Additional enhancements to the NeRF-style networks were made by NeRF– [58], which eliminate the need for known camera parameters; they estimate the camera parameters through joint optimization during training, reporting at-par and sometimes better results over Colmap-based pose estimation baselines. Neural Sparse Voxel Fields [59] reported a speed-up of over 10 times over NeRF at inference along with a better quality of rendering, by using a sparse voxel octree structure to represent local properties in each cell of a voxel-bounded implicit field, and skipping irrelevant scene content (as learned during ray marching) at the time of rendering. SIREN [60] generalized the periodic activation function concept of NeRF to multilayer networks, allowing detailed representation of images, audio, and video. In recent research, papers such as DefTet [61] and [62] address the inadequacies of NeRF's volume density estimation, and propose approaches for learning deformable meshes for reconstruction.

Due to the recent explosion of research in neural rendering, we do not cover many papers that have contributed to the efficiency and popularity NeRF-style networks see today; however, an excellent review is available at https://dellaert.github.io/NeRF/. We discuss dynamic NeRF papers at a later stage in the thesis, as foundation for spatio-temporal reconstruction using Neural Volume Rendering methods.

## 5.3 Approach

### 5.3.1 Neural Radiance Fields

Neural Radiance Field, or NeRF, is a method used to represent scenes using a fully connected MLP, while being less expensive storage-wise than discretized voxel grids. The scene to be rendered is represented using a continuous 5D function of location in (x,y,z) and orientation $(\theta, \phi)$. The 5D coordinates are queried along rays marched through the

scene to form the input to the neural net, and regressed to a single volume density and color for each ray. The method exploits its differentiability to use gradient descent optimization by minimizing errors between multiple observed and rendered views. The Figure 5.1 from the paper indicates the pipeline.

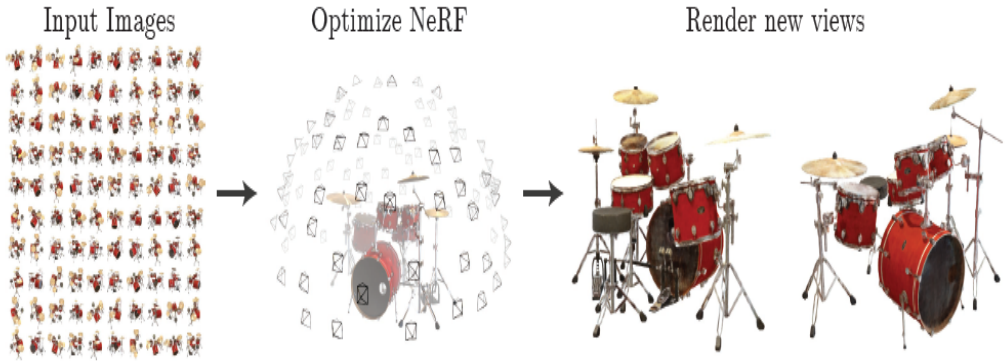



Figure 5.1: NeRF's data capture and rendering flow

The scene representation uses only the location (x,y,z) to predict the volume density along the ray, while the RGB color prediction is dependent on both, the viewing direction (represented in practice as a 3D Cartesian unit vector **d**), and the location.

Since deep neural networks have been shown to be biased towards learning low frequency features [63], the rendered representations generally fail in accurately capturing high frequency features. In order to represent high frequency functions, NeRF maps the input coordinates to a higher dimensional space via a positional encoding (periodic activation functions).

It is to be noted that NeRF uses the same coordinate system as OpenGL, with the local camera coordinate system defined with +X axis to the right, +Y axis upwards and +Z axis backwards from the image. The poses are represented by 3x4 camera-to-world transformation matrices.

### 5.3.2 NeRF Synthetic Dataset

Our data capture methodology so far has been partially informed by the pandemic, and its effects on the hydroponic growth experiments, our image capture frequencies and pipelines at the Georgia Institute of Technology. Due to the unique input views of NeRF, we were unable to capture images using a real lettuce to assess the performance of radiance fields in rendering and quantifying the volume of the lettuce. To work around this, we created a synthetic dataset in Blender with a Python script, with a randomized number of leaves and layers around a sphere that simulates the lettuce head. The script for the lettuce was based off of the Clevr Dataset Gen [64]. Of the available material blend files from Clevr (matte and metal), we use matte along with varied hues of green and yellow to simulate the leaf texture and color, and blend files of basic shapes were combined to form lettuces, such as a sphere for the lettuce head and a few varieties of manually sculpted leaves combined in random patterns and numbers to form the lettuce. We added constraints in the placement of the objects to make the lettuce as realistic as possible. A single sphere was consistently placed in the middle of the scene, and the leaves were spatially arranged around the sphere, with the base of the leaves close to the base of the sphere in order to simulate a lettuce, as shown below. For multiple rows of leaves, we enforced a tilt angle and size mappings, as a function of the row, to realistically model the leaves at the bottom (in a real lettuce, the first grown leaves) as being flatter and bigger than the newly sprouted leaves at the top of the lettuce. Within each row, we select the angular position of the leaf around the Z-axis (Blender having a right handed coordinate system with Z running from top to bottom) such that the leaves do not fully overlap. In order to capture multiview images from the Blender lettuce, we then use the camera poses used in NeRF's synthetic dataset (as given in the examples file hosted on the repository. These camera poses are extracted and converted to azimuth and elevation angles, which are used by Blender to position the camera around the scene.

For each scene, we capture around 100 images for training and validation each, and 200

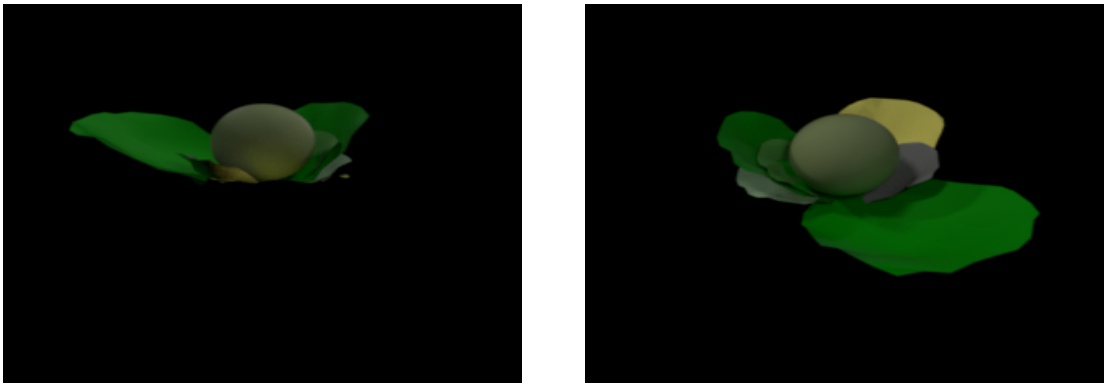


Figure 5.2: Lettuce simulated on Blender

images for testing, with cameras positioned on a semi-sphere around the lettuce.

Given a synthetic dataset's black background, NeRF uses a 4-channeled dataset to query only foreground regions in the image. For our dataset, we added an alpha channel that marked every black pixel as transparent. We trained NeRF on this synthetic dataset using a GPU with 1080Ti, 11Gb of vRAM, with the training period coming to approx 12-15 hours.

Figure 5.3: Lettuce simulated on Blender

### 5.3.3 NeRF Real Dataset

In order to test NeRF's performance on real world applications, we printed a 3D model of a synthetic lettuce rendered using Blender, and converted it to a watertight mesh using Meshlab. We choose the printed model over capturing images of a real lettuce due to the possibility of computing the ground truth volume through software, thus enabling accurate

evaluation of the results of volume estimation from the rendered model. This printed model was placed on a white disk with marker tags around the disk, and the lettuce at the center. A video of the lettuce was captured using a 30 fps Intel Realsense d435 camera, saved as a rosbag file. The images were extracted from the rosbag using its rectified topic, and filtered to remove blurry images using OpenCV's blur detection algorithm (which computes blur as the variance of the Laplacian of the image). The remaining images were then fed to LLFF [65] to extract the poses in NeRF-requisite format, from images.

*LLFF*

Local Light Field Fusion (LLFF) is an approach to novel view synthesis using sparse input images. They use an irregular grid of input images that use an MPI scene representation to expand images into a local light field; and novel views are rendered by interpolating adjacent light fields.

Designed to synthesize new views from a local layered representation, we use their code to estimate camera poses for our dataset. The code is a wrapper over Colmap [66], to get 6 DoF camera poses and depth bounds of the scene, and converts the Colmap results to a format directly usable by NeRF. For real datasets, we only need to pass in the scene directory and the pose information generated from LLFF to NeRF, which then automatically partitions the data.

The poses estimated from Colmap are saved in a numpy array of size Nx17; with N being the number of images. Each row can be reshaped into a 3x5 pose matrix, and the remaining two values represent the closest and furthest depth bounds of the scene in the context of the local view. As mentioned before, the poses are a 3x4 camera-to-world transformation, with the last 3x1 column representing camera intrinsics of the format $[height, width, focal\_length]^T$. Here we assume that there is a single focal length in both the X and Y directions, and that the principal point is centered. As part of post processing the Colmap results, the code also converts the poses from Colmap's coordinate

system (X, -Y, -Z) corresponding to right of the image, forward and downwards, to (-Y, X, Z) corresponding to downwards, right, and backwards from the image.

In order to speed up NeRF and aid in volume estimation, we attempted to segment out the background in order to render only the object of the interest, making it easier to estimate volume from the rendered and triangulated mesh. To effectively segment the lettuce with varying hues due to lighting changes and a cluttered background, we compute a color histogram of a manually selected ROI, and measure the hue and saturation values. Setting the threshold at 85 percentile, we segment clusters of areas containing the shades of green present in the lettuce, as well as some. We then filter out the precise lettuce area by retaining the largest contour in the segmented image. We then run these segmented images along with their poses through NeRF, resulting in the rendering shown in fig Figure 5.4.

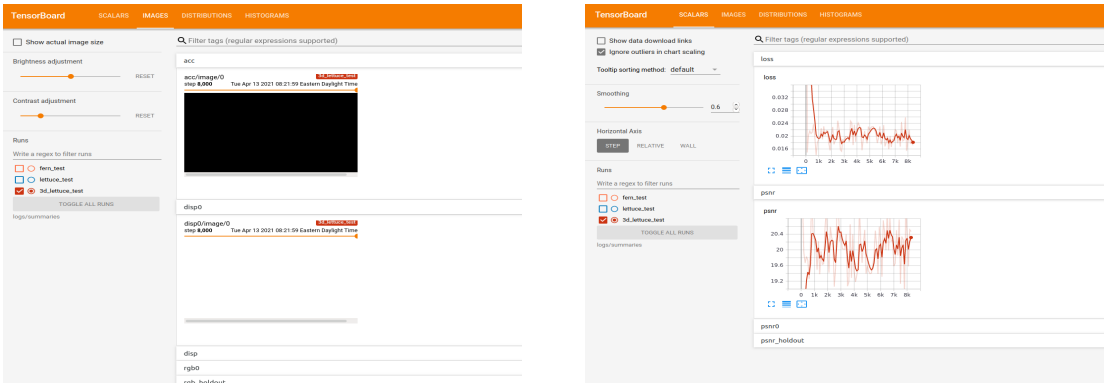



Figure 5.4: Tensorboard viz of NeRF's learning

As can be observed, NeRF does not learn anything during the training phase. This is due to the data processing being different in LLFF data than Blender data. While Blender required a mask to learn only the object in the scene by rendering rays only at points designated as foreground, LLFF uses the near/far bounds to regress the color and density values along the ray. For our initial dataset (shown in Figure 5.5) that was captured in a background-inclusive, inward facing camera trajectory, the resultant far bound was at near infinity. Additionally, the lettuce covered less than 60% of the scene. When fed segmented data, the ray through the pixels of the lettuce regress with the color green for part of the ray,

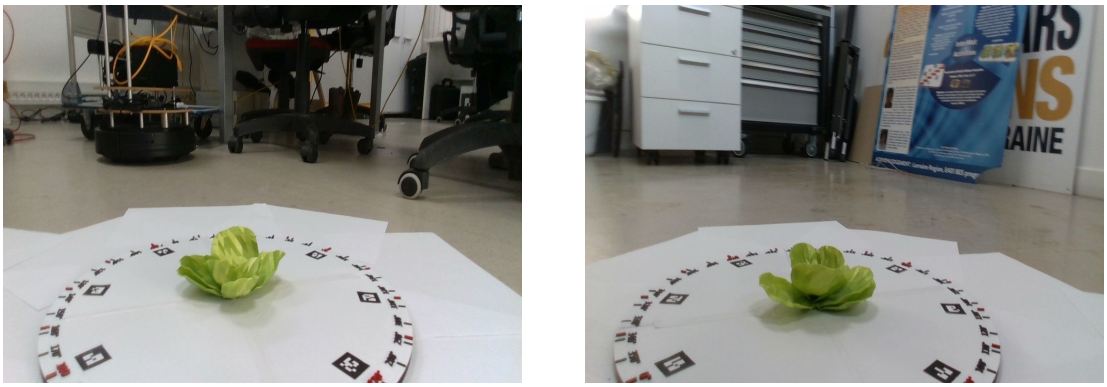
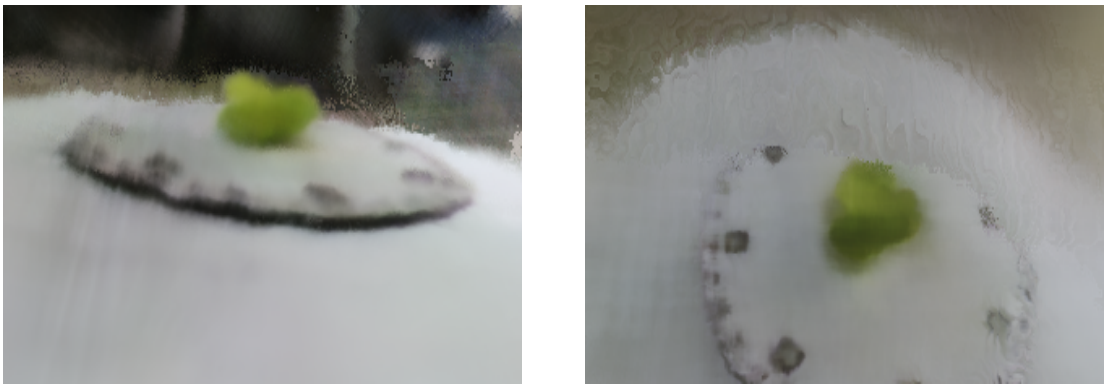


Figure 5.5: Sample images from 3D printed lettuce

then black for the area 'behind the lettuce', so to speak. NeRF renders rays by randomly sampling from within a subsampled area- and in such cases, our reasoning is that a majority of the rays sampled would be black, with the green parts of the ray considered noise. To test this hypothesis, we use the images with the background kept in, as input to NeRF. After a few thousand iterations, it results in better rendering outputs, as shown in fig Figure 5.6. As can be seen, the images are diffuse- possibly due to a large portion of the capacity being used for learning the background. As observed, the PSNR and loss also seem a lot more



(a) at 5000 iterations (b) at 12500 iterations

Figure 5.6: NeRF's learning with background

coherent and conform to the expected curves. Compared to using a segmented dataset, this approach adds additional compute and overhead to segment the lettuce mesh from the scene to estimate the volume.

However, running trials with a dataset looking down towards the lettuce at an incline,

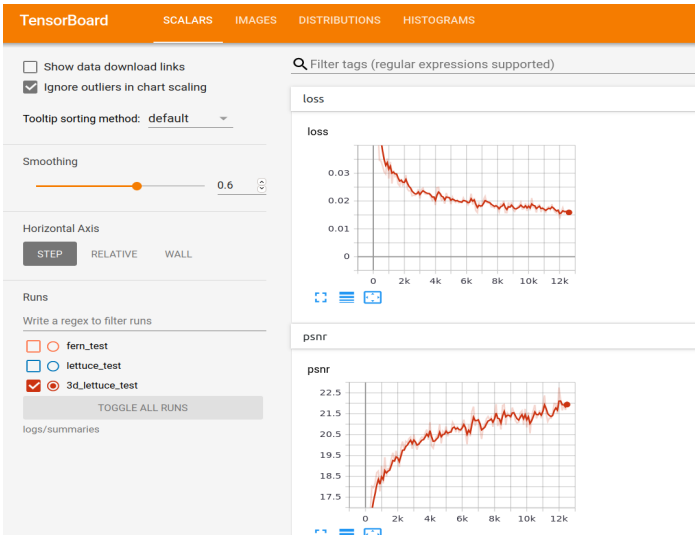



Figure 5.7: Metrics at 12500 iterations

and zoomed in so the lettuce would fill a majority of the scene, also did not render the expected results, with NeRF still not learning a representation. We are still investigating possible reasons for this phenomena.

Another approach could be to clip the far bound values returned by the LLFF data. This could potentially only regress the ray as far as the lettuce depth, resulting in accurate rendering and helping make volume estimation easier. The trick here would be to correctly compute the value that the far bound should be clipped to- an empirical method could be to visualize the 3D rays and manually estimate an approximate depth; however, this is time-consuming, approximate and requires manual intervention.

Unlike synthetic data, running NeRF on non-synthetic data does not need separating into train, test and val folders. As input, we provide the segmented images and the file containing camera poses, intrinsics and scene depth bounds for each image.

One of the issues we face here, and which will be discussed later in the Discussions section, is the lighting variability when capturing images of the lettuce. The printed lettuce has slightly reflective surfaces, and NeRF bakes in lighting, which affects the learning of color surfaces from different views.

## 5.4 Summary

Here, we address neural volume rendering as a highly promising approach towards 3D reconstruction. We describe the data capture method we use, and the difference in process between synthetic and real world capture and learning. In the next chapter, we analyze the results in a real world setting, and discuss best practices for data capture.

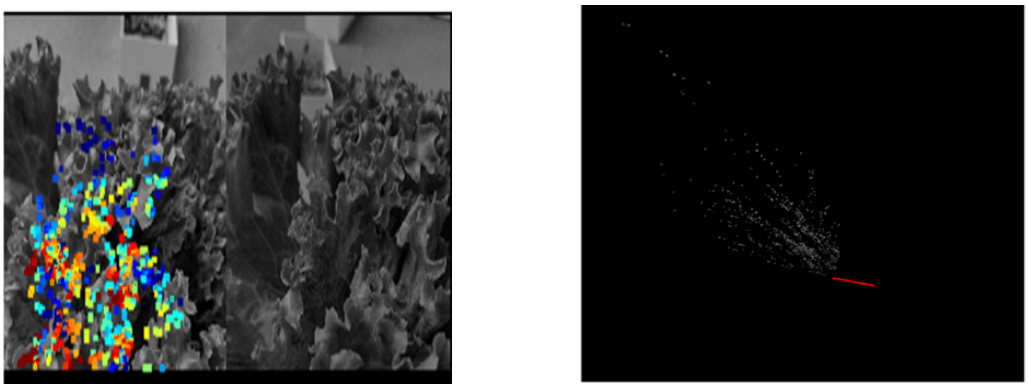
# CHAPTER 6

# RESULTS AND DISCUSSION

## 6.1 Survey of Different Solutions - photogrammetry softwares

Here we present results from some of the other approaches we evaluated lettuce datasets against:

1. DSO: We ran a sample lettuce dataset, taken without the robot arm on the DSO code downloaded and built from https://github.com/JakobEngel/dso. While the color coded depth map seemed in line with the SIFT-based feature extraction, the SFM results were very poor (See Figure 6.1). Given our data collection methods, we attribute the results to the following factors: 1. Accurate geometric and photometric calibration, and 2. the data collection trajectory of the robot arm. The former factor has not been extensively tested, ie. we have not compared the performance between datasets collected with and without the robot arm, and hence is simply a speculation.



(a) Detected points via DSO (b) DSO SFM result

Figure 6.1: DSO Output

2. Metashape: Metashape is a commercial photogrammetry tool that was reviewed in

[67]. It is a successor of the Agisoft Photoscan software, and uses a SIFT-like feature extraction algorithm. While solving for camera parameters, it uses a greedy algorithm and bundle adjustment to find and refine camera poses. The dataset was run on a demo mode of the product. These gave comparatively good results, but still exhibited messy edges.

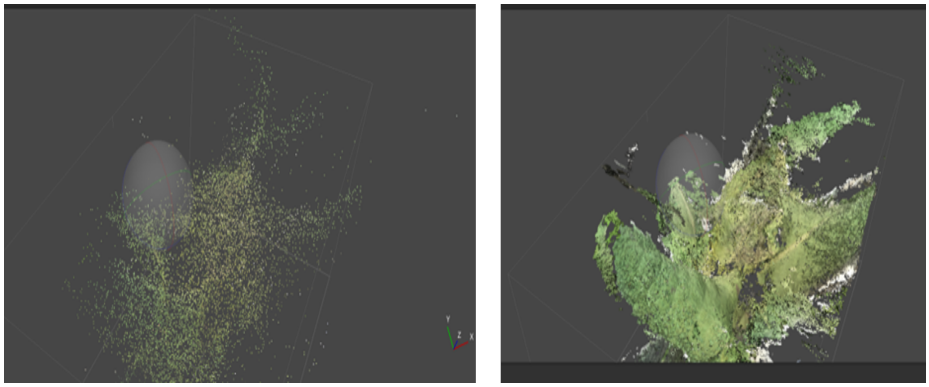


Figure 6.2: Sparse and dense reconstruction from Metashape

3. Regard3D: This is also a commercial photogrammetry software, an open-source SFM pipeline. It uses AKAZE and an incremental bundle adjustment for the reconstruction process. Compared to Metashape, this gives better results; which is possibly due to the incremental pipeline, as opposed to Metashape's global bundle adjustment.

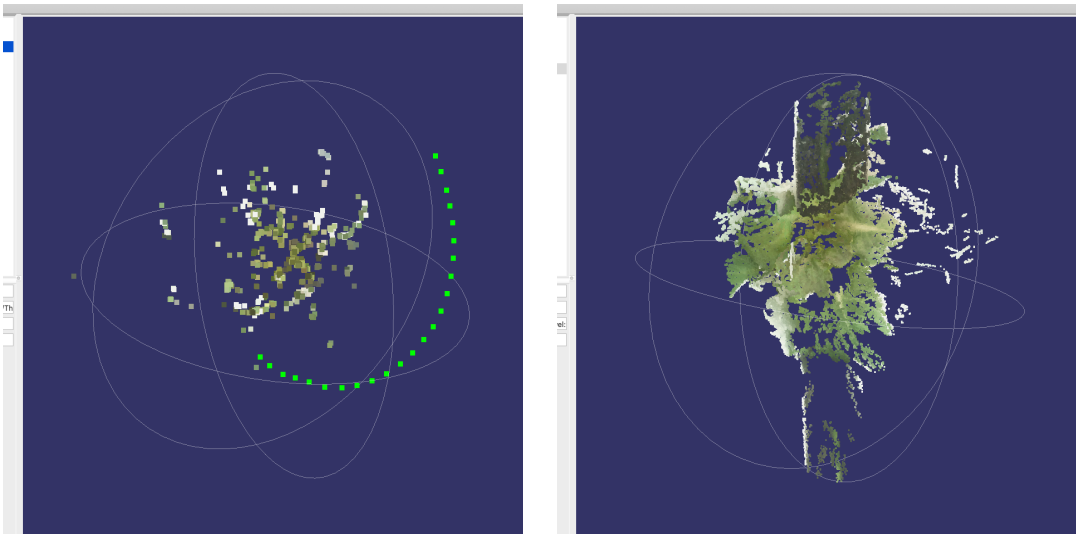


Figure 6.3: Sparse and dense reconstructions using Regard3D

4. Colmap: The current state-of-the-art in image reconstruction, Colmap uses an incremental BA pipeline. While accurate and robust, this is a very time-intensive process for large datasets.

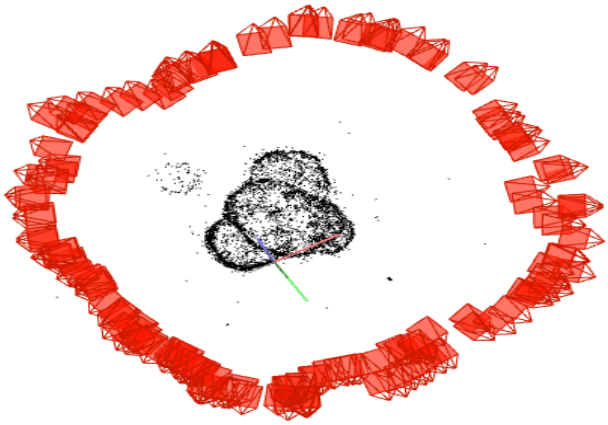


Figure 6.4: Colmap Sparse Reconstruction

## 6.2 Solutions from Thesis approaches

In this section, we analyze volume estimation from the different approaches explained in previous chapters. In order to analyze accuracy, we use a synthetic lettuce with known volume as ground truth. The lettuce was designed on Blender, converted into a watertight mesh in an STL format using Meshlab, and 3D printed using PrusaSlicer. The volume of the model was also computed from the 3D printing software, as 11282.62, with the corresponding printed model dimensions being 100x82.5x46.5 $mm^3$. This computed volume is unitless as the software cannot retrieve the physical scale of the model. We then extrapolate the performance of these algorithms with real lettuce, given controlled environmental and image capture conditions. We explain how the volume was calculated, along with an evaluation of accuracy and tradeoffs in different methods.

### 6.2.1 LAI

We observed an $R^2$ value of 0.94 on validation set for wet weight estimation. We posit that the estimation accuracy can be improved, even if slightly, by integrating multiple views for better LAI computation, as opposed to the current computation using only front facing images. While this method gives a satisfactory rough estimate of wet weight, enabling biomass computation, no phenotype change is recorded or monitored. In addition, this method requires significant manual effort and is inefficient and expensive, which makes it unsuitable for scaling to a chamber-setup with many dozen plants.

While direct volume estimation is not possible with this approach, we approximate a proportional relationship between mass and volume, and use this to roughly predict the biomass and nutrient uptake of the plant for different solutions.

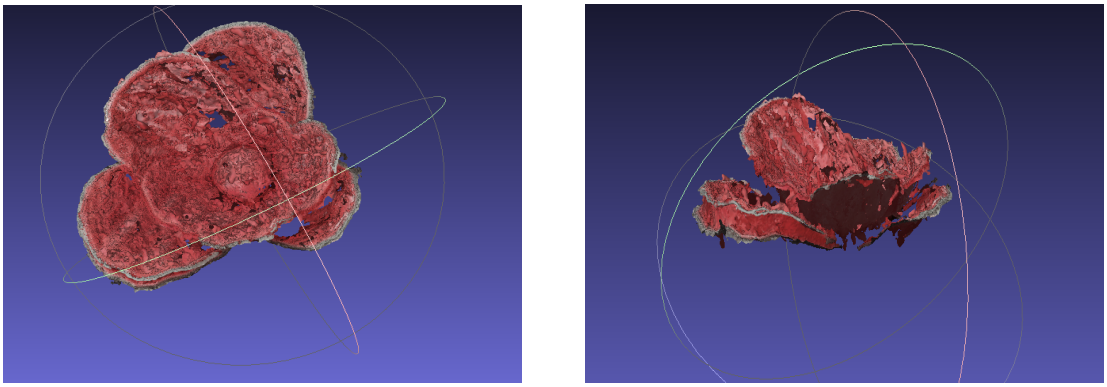
### 6.2.2 Curve Reconstruction

Given the superior results of colmap's reconstruction over ours, and their built-in functions to compute a dense point cloud and subsequently a colored mesh, we evaluate the mesh computed from colmap as input to volume estimation. Below in Figure 6.5, we show rendered mesh output from colmap, for a single incline of images (similar to NeRF). As we can see, the hollow structure and the color of the lettuce is captured very well, however, the mesh has multiple holes, and is disjoint due to inadequate depth information.

Attempting to fit a Screened Poisson surface reconstruction [68] to this mesh results in a plane being fit over the scene, as shown in Figure 6.6. Due to the disjoint nature of the mesh, volume estimation is not possible without additional images for reconstruction.

### 6.2.3 NeRF

For the data capture for NeRF, we use an inclined top-down facing camera trajectory to capture images of the printed lettuce in a spherical fashion, similar to SfM, but without multiple inclinations. Doing so restricts the depth of the scene, while providing enough



(a) topview (b) side view

Figure 6.5: Poisson reconstruction by Colmap

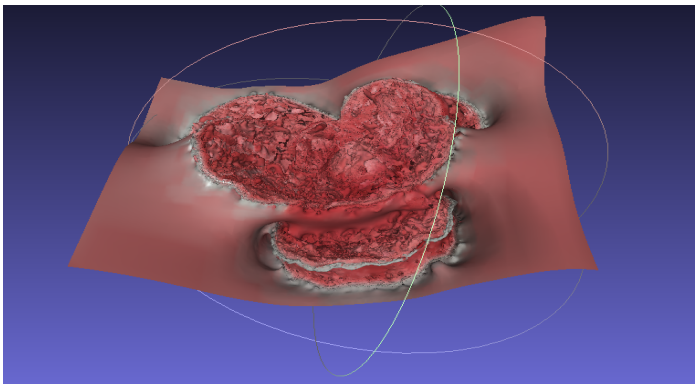
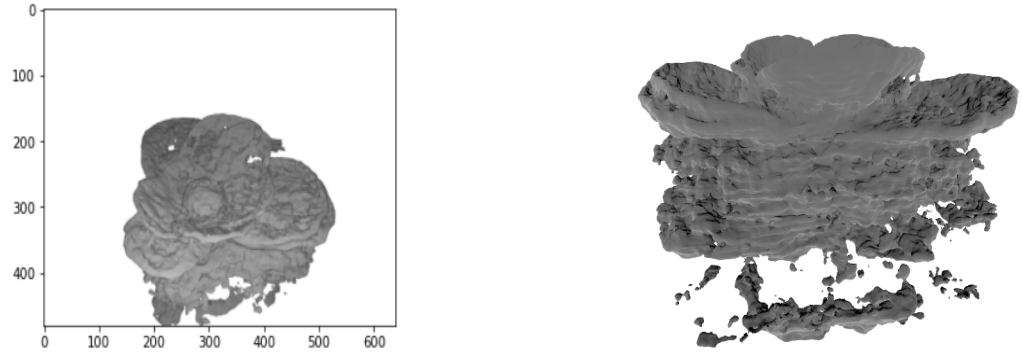


Figure 6.6: Screened Poisson on Colmap output

details and lesser background for the scene to render.

In order to use the rendered model as input for volume estimation, we require to convert the model into a triangulated mesh. We do this using the marching cubes algorithm [69]. Simply put, marching cubes creates a polygonal mesh from an implicit function by iterating over a cuboid-shaped area of 8 neighbor locations superimposed over the localized regions. If the surface passes through the cube, the algorithm computes the polygon that best represents the surface through the triangles generated from the vertices intersecting the cube. The final mesh is the result of fusing these localized triangles in their appropriate positions. NeRF's authors provide a notebook that uses the PyMCubes package to extract the mesh from a trained model. We use Meshlab [70] to process the rendered mesh, by removing artifacts caused due to the periodic activation function, closing holes and making the model

watertight through a Poisson reconstruction [71]. Meshlab then scales the model into a hypercube, and computes the volume of the mesh with respect to the hypercube, making it essentially unitless. Figure 6.7 shows the rendered mesh extracted from NeRF. We note that



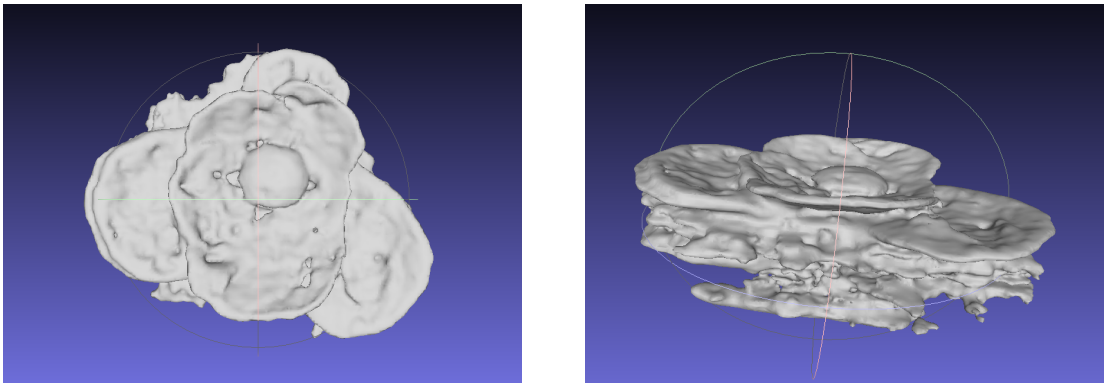

(a) topview (b) side view

Figure 6.7: NeRF rendered mesh

while the top of the lettuce has been rendered with reasonable accuracy, the depth of the lettuce is inaccurate. This is due to the image capture not adequately capturing the depth of the lettuce, leading the network to make assumptions. This can be addressed by incorporating additional images at different inclines from the ground plane, thus adding depth information - however, as the images were taken from a handheld camera in a home setup, inclines near the ground plane have visible cluttered background, introducing artifacts and decreasing the network capacity, as discussed later in this section. While assumptions can be made regarding the effectiveness of the network with background removed or under controlled setups, it is a non-trivial technique and out of the scope of this thesis.

Another issue we note here is that the rendered mesh cannot be directly used for volume estimation; and the surface reconstructed mesh loses texture information. However, we surmise that this could be due to NeRF having run on too few iterations- in this case, it was run for only 40k steps, while the recommended duration is 200k steps.

The watertight surface reconstruction obtained from mesh postprocessing in Meshlab estimates the volume as 22898.14, nearly 2x the ground truth volume. We theorize that the error could be due to the solid block of incorrect depth under the rendered lettuce, as



(a) topview (b) side view

Figure 6.8: Meshlab's surface reconstruction of the rendered mesh

opposed to the relatively hollow structure of the printed lettuce. We provide Figure 6.9 as an illustration of this point. As mentioned, this could potentially be alleviated by incorporating more images at different inclines and cleaner background, but evaluating this hypothesis was not possible with the currently available dataset.

In order to scale the volume to physical units, we can use the rendered marker of known size, rendered in the background of the scene (not shown in the above images) to estimate the relation between Meshlab's volume and the real volume of the object.

We also note that the generated mesh does not incorporate color, an integral part of plant monitoring- however, a number of approaches, such as [72] render 360 inward facing scenes such as ours and provide scripts to extract colored mesh. They do this by projecting vertices on to the training images to obtain rgb values, while computing opacity along rays in the NeRF network to estimate view-based occlusion. The provided notebooks in the repository make getting the colored mesh a very straightforward procedure, from an application standpoint.

In the absence of data capture for real lettuce, in order to map the volume and phenotypes over time, we can render multiple lettuces at different stages in their lifecycle, such as on Blender, and use that to plot the growth over time. Given the accuracy of the volume estimation for a simulated lettuce, we can confidently state that this approach will work at all stages of a lettuce. This can similarly be extrapolated towards mapping growth information

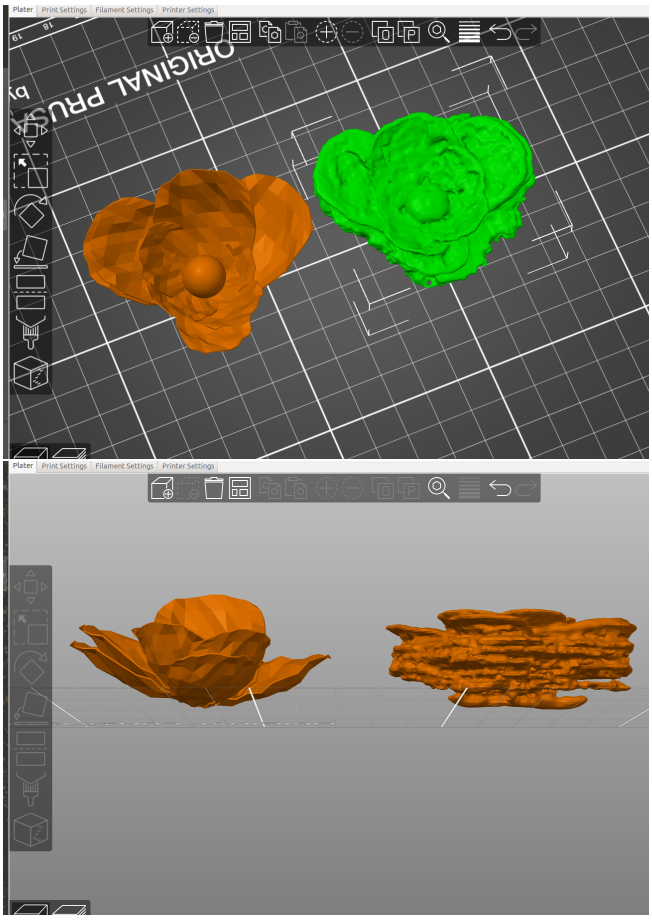


Figure 6.9: Comparison of printed model (left) and rendered mesh (right)

at different stages for a real lettuce.

One of NeRF's biggest drawbacks is the inability to generalize learning to new geometries. The time taken to train each individual lettuce geometry in order to synthesize new views is prohibitively expensive, and unsuitable for real-world applications where multiple lettuces need to be processed at frequent intervals. However, a Pytorch version of NeRF was developed by Yen-Chen Lin et. al, which runs up to 1.3 times faster [73]. Further speed improvements, when developed, may change neural rendering to be the technology of choice in actual applications, given their high accuracy.

Table 6.1 provides the subjective results of different approaches over parameters such as reconstruction quality (for visual monitoring), computational cost and finally, estimated volume.

Table 6.1: Approaches and observations

| Approach | Computational Cost | Reconstruction Quality | Volume Estimated |
|---|---|---|---|
| Curve Reconstruction | Low | N/A | N/A |
| Colmap | Medium | Medium | N/A |
| NeRF | Very high | Medium | 22898.14 |

## 6.3 Discussion

### 6.3.1 SfM

For the sake of proof-of-concept, we use a 3D printed lettuce to evaluate an SfM reconstruction. However, we can extrapolate the results to a real lettuce, captured with an RGB camera mounted to a robot arm. While we have used a mobile camera to capture the final dataset for evaluation of all 3 approaches, we preprocess the data in order to simulate as realistic a scenario as possible for the curve reconstruction algorithm, for example by providing pose priors from poses estimated via other methods such as GTSFM. The main challenges with reconstruction via SfM has been the repeated texture of the lettuce, and the algorithm's convergence dependence on the dataset collection. A few points to be noted when capturing data is:

- Do not crop the lettuce images. Care should be taken to have high-quality zoomed out images that, while containing enough details within the plant, also have a large portion of the plant (and neighborhood curves) visible.
- Care should be taken to avoid/remove image distortion effects. While distortion such as radial can be easily removed given known intrinsics, cameras and capture methods should be chosen so as to avoid effects like Depth-of-Field.
- The problem can be reduced to simpler terms by positioning markers around the lettuce, such that the markers can be used as 'anchors' to ensure accurate correspondences. This could not be done with the current dataset due to the pandemic.

- While other, more efficient, approaches to reconstructing lettuce exist (such as using a laser scanner, or a depth camera), we use readily-available cameras and data capture methods; partially as an interesting research project, and partially due to the scarcity of resources due to the pandemic. However, these approaches might lend themselves well to reconstruction evaluation on a proper hydroponic setup post-pandemic.

Additionally, SfM is sensitive to noise - we note the efficiency of neural rendering approaches in converging even with noisy data, such as with blurred images and with lesser data; whereas SfM fails to converge in such scenarios.

### 6.3.2 NeRF

For real life datasets captured for NeRF, it is highly recommended to enforce as plain a background as possible. It has been observed that for spherical scenes, background changes between views introduced noisy artifacts in the output, as shown in fig Figure 6.10.

Figure 6.10: Rendering artifacts due to cluttered background

Other settings that could improve results are ensuring constant exposure across all images, as effects like specular variation cause issues with learning the color of the object, and render scenes with significant artifacts. However, given the environment controlled, windowless chamber used in hydroponics, it might be easier to control these effects during a real-world capture. Another point to be noted is that for the current configuration of the chamber and arm, where the plant is almost perpendicular to the robot arm base, it is worth

studying the possibility of taking a true LLFF-style data capture in a grid based pattern, as opposed to a spherical pattern, and its effects on the depth, and subsequently volume, estimation.

## 6.4 Future work: 4D Association

As noted, we do not perform temporal association within the scope of this thesis. However, multiple papers exist in the research domain for true temporal association for a particular approach.

### 6.4.1 SfM

To associate point clouds over time, the first step would be registration, the process of finding a geometrical transformation that pairwise aligns point clouds. Iterative Closest Point (ICP) is the most popular algorithm used to match point clouds, which, as the name suggests, iteratively refines the transformation required to minimize the distance between points in the reference and source clouds. However, this approach is susceptible to outliers and accuracy of initial matches for aligning. [74] attempts to increase robustness by using center of gravity as reference points and a point-pair distance constraint. [75] addresses the registration problem using hue data, combining normalized point range and weighted color values from the associated images. However, given the symmetry and uniform color over our dataset, the effectiveness of this approach would need to be evaluated. [76] describes the PointNet algorithm as a learnable function, integrating Lucas-Kanade and PointNet into an RNN for robust point cloud alignment.

The main drawback with the above approaches is that they do not account for dynamic point clouds, ie. plant growth over time; making them very difficult to be used for registering crops of different sizes and performing temporal association. The most relevant papers for us, perhaps, are [77] and [78], that are directed towards crop registration and association; albeit for simpler crops than the lettuce.[78] proposes a point cloud registration

method designed to tackle growth, changing structure and non-rigid motion between temporal point cloud instances. Using Hidden Markov Models (HMM), they estimate skeletal correspondences for each point cloud pair, and compute deformation parameters through a non-linear optimization procedure. These deformation parameters are also shown to be useful in interpolating point clouds between temporal instances. [79] describes Fast Point Feature Histograms- robust local feature descriptors for point clouds, based on each point's localized neighborhood. [77] builds on this by computing keypoints using FPFH and SVM to achieve semantic segmentation and unsupervised clustering, and using the semantic information to inform their data association and non-rigid registration processes. They report performance improvements over [78], and excellent results on day-over-day leaf area computation with challenges such as new plant growth, leaf bending and missing data.

### 6.4.2 NeRF

Dynamic NeRF is a far more difficult problem than 4D SfM, due to the learning constraints and time consumption required for training single scenes. Being a relatively nascent field, research in the domain of extending NeRF to dynamic scenes is fairly new, with fewer methods than SfM based temporal reconstruction approaches.

[80] and [81] consider time as an input to the network, and train their networks in 2 simultaneous stages: encoding the scene in a canonical space (like NeRF), and using a secondary MLP to map the scene to a deformation to synthesize novel views at time $t$. [82] is specialized for 4D monocular facial reconstruction, and integrates a low-dimensional morphable model into the SRN, providing pose and expression control. However, the strong shape prior makes it unsuitable for dynamically changing geometry such as observed in lettuce. While the above papers work on the assumption of a base 3D structure, over which some deformation is applied, we are more interested in deformable object rendering. The two papers that seem closest to achieving that objective are [83] and [84]. [84] learns spatio-temporal representation of a dynamic scene by learning the underlying 3D structure

of the scene and representing flow directions as RGB coordinates; resulting in multiview rendering capability across different time steps. [83] represents a dynamic scene as a function of motion, geometry and appearance over time. Their space-time view synthesis shows non-rigid deformation modeled over fixed time, interpolated view and fixed view, interpolated time paradigms. Of these, I believe the fixed view, interpolated time rendering might be especially useful for the nature of plant deformation and temporal monitoring we can expect in the application domain.

## REFERENCES


[1] A. Strothkämper, "Farm to fork goes digital: How agribusiness digiti- zation can feed the world," 2016.

[2] R. Sharma, S. Kamble, A. Gunasekaran, V. Kumar, and A. Kumar, "A systematic literature review on machine learning applications for sustainable agriculture supply chain performance," *Computers  Operations Research*, 2020.

[3] R. R. Shamshiri, F. Kalantari, K. C. Ting, K. R. Thorp, I. A. Hameed, C. Weltzien, and et al, "Advances in greenhouse automation and controlled environment agriculture: A transition to plant factories and urban agriculture," in *International Journal of Agriculture  Biological Engineering*, 2018.

[4] A. Both, "Ten years of hydroponic lettuce research," *Self-Review*, 2020.

[5] Z. Rengel, "Mechanistic simulation models of nutrient uptake: A review," *Plant and Soil*, 1993.

[6] J. Monod, "The growth of bacterial cultures," *Annual Review of Microbiology*, 1949.

[7] F. Fiorani and S. Ulrich, "Future scenarios for plant phenotyping," *Annual Review of Plant Biology*, 2013.

[8] A. Tzounis, K. Nikolaos, B. Thomas, and K. Constantinos, "Biosystems engineering," 2017.

[9] M. Alipio, A. Cruz, J. Doria, and R. Fruto, "On the design of nutrient film technique hydroponics farm for smart agriculture," *Engineering in Agriculture, Environment and Food*, 2019.

[10] E. D. Pocholo James M.Loresco Ira C. Valenzuela, "Color space analysis using knn for lettuce crop stages identification in smart farm setup," in *TENCON, IEEE Region 10 Conference (Jeju, Korea)*, 2018.

[11] R. Vicerra, P. Loresco, and E. Dadios, "Segmentation of lettuce plants using super pixels and thresholding methods in smart farm hydroponics setup," Jul. 2019.

[12] P. J. Loresco, I. Valenzuela, A. Culaba, and E. Dadios, "Viola-jones method of marker detection for scale-invariant calculation of lettuce leaf area," in *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*, 2018.

[13] D. Watson, "Comparative physiological studies on the growth of field crops: I. variation in net assimilation rate and leaf area between species and varieties and within and between years," *Annals of Botany*, 1947.

[14] A. Iio, K. Hikosaka, N. P. R. Anten, Y. Nakagawa, and A. Ito, "Global dependence of field-observed leaf area index in woody species on climate: A systematic review," *Global Ecology and Biogeography*, 2014.

[15] D. M. Firman and E. J. Allen, "Relationship between light interception, ground cover and leaf area index in potatoes.," *The Journal of Agricultural Science 113, no. 3*, 1989.

[16] A. Gregory, J. Scurlock, and J. Hicke, "Global synthesis of leaf area index observations: Implications for ecological and remote sensing studies.," *Global Ecology*, 2003.

[17] V. Kinhal. (). "The importance of leaf area index (lai) in environmental and crop research."

[18] S. M. Weraduwage, J. Chen, F. C. Anozie, A. Morales, S. E. Weise, and T. D. Sharkey, "The relationship between leaf area growth and biomass accumulation in arabidopsis thaliana," *Frontiers in Plant Science*, vol. 6, p. 167, 2015.

[19] *Photoscissors*.

[20] S. Delagrange and P. Rochon, "Reconstruction and analysis of a deciduous sapling using digital photographs or terrestrial-lidar technology," *Annals of Botany*, 2011.

[21] R. Klose, J. Penlington, and A. Ruckelshausen, "Usability study of 3d time-of-flight cameras for automatic plant phenotyping," *Bornimer Agrartechnische Berichte*, 2009.

[22] K. Omasa, F. Hosoi, and A. Konishi, "3d lidar imaging for detecting and understanding plant responses and canopy structure," *Journal of Experimental Botany*, 2007.

[23] R. Flavel, C. Guppy, M. Tighe, M. Watt, A. McNeill, and I. Young, "Non-destructive quantification of cereal roots in soil using high-resolution x-ray tomography," *Journal of Experimental Botany*, 2012.

[24] P. Gregory, D. Hutchison, D. Read, P. Jenneson, W. Gilboy, and E. Morton, "Non-invasive imaging of roots with high resolution x-ray micro-tomography," *Plant Soil*, 2003.

[25] U. Rascher, S. Blossfeld, F. Fiorani, S. Jahnke, M. Jansen, and et al., "Non-invasive approaches for phenotyping of enhanced performance traits in bean," *Functional Plant Biology*, 2011.

[26] H. Poorter, J. B¨ uhler, D. van Dusschoten, J. Climent, and J. Postma, "Pot size matters: A meta-analysis of the effects of rooting volume on plant growth," *Functional Plant Biology*, 2012.

[27] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision.*, 1999.

[28] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571.

[29] A. B. Pablo Alcantarilla (Georgia Institute of Technolog) Jesus Nuevo (TrueVision Solutions AU), "Fast explicit diffusion for accelerated features in nonlinear scale spaces," in *Proceedings of the British Machine Vision Conference*, BMVA Press, 2013.

[30] K. Wilson, D. Bindel, and N. Snavely, "When is rotations averaging hard?," vol. 9911, Oct. 2016, pp. 255–270, ISBN: 978-3-319-46477-0.

[31] R. Hartley and P. Sturm, "Triangulation," *Computer Vision and Image Understanding*, 1997.

[32] S. Xie and Z. Tu, "Holistically-nested edge detection," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1395–1403.

[33] S. Belongie, J. Malik, and J. Puzicha, "Shape context: A new descriptor for shape matching and object recognition," in *NIPS*, 2000.

[34] S. Belongie and J. Malik, "Matching with shape contexts," in *IEEE Workshop on Contentbased Access of Image and Video Libraries*, 2000.

[35] S. B. J. M. J. Puzicha, "Shape matching and object recognition using shape contexts," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[36] H. W. Kuhn and B. Yaw, "The hungarian method for the assignment problem," *Naval Res. Logist. Quart*, pp. 83–97, 1955.

[37] A. Baid, F. Jiang, A. Krishnan, J. Lambert, A. Singh, A. Venkataramanan, S. Warrier, J. Wu, X. Wu, and F. Dellaert, *Gtsfm: Georgia tech structure from motion*, https://github.com/borglab/gtsfm, 2021.

[38] F. Dellaert, D. Rosen, J. Wu, R. Mahony, and L. Carlone, "Shonan rotation averaging: Global optimality by surfing so(p)$^n$," in *ECCV*, 2020.

[39] K. Wilson and N. Snavely, "Robust global translations with 1dsfm.," in *ECCV (3)*, D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., ser. Lecture Notes in Computer Science, vol. 8691, Springer, 2014, pp. 61–75, ISBN: 978-3-319-10577-2.

[40] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[41] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. USA: Cambridge University Press, 2003, ISBN: 0521540518.

[42] M. Kesson, *Vectors: Shortest distance from a point to a line*, 2002.

[43] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," 2012.

[44] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, ser. Adaptive computation and machine learning. MIT Press, 2009, ISBN: 9780262013192.

[45] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B. Goldman, and M. Zollhöfer, "State of the art on neural rendering," *EG*, 2020.

[46] M. M. Ozguven and K. Adem, "Automatic detection and classification of leaf spot disease in sugar beet using deep learning algorithms," *Physica A: Statistical Mechanics and its Applications*, 2019.

[47] A. Kamilaris and F. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018.

[48] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[49] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[50] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[51] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2019.

[52] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe, *Deep local shapes: Learning local sdf priors for detailed 3d reconstruction*, 2020. arXiv: 2003.10983 [cs.CV].

[53] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[54] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, "Neural volumes: Learning dynamic renderable volumes from images," *ACM Trans. Graph.*, vol. 38, no. 4, 65:1–65:14, Jul. 2019.

[55] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhöfer, "Deepvoxels: Learning persistent 3d feature embeddings," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

[56] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene representation networks: Continuous 3d-structure-aware neural scene representations," in *Advances in Neural Information Processing Systems*, 2019.

[57] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[58] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, "NeRF−−: Neural radiance fields without known camera parameters," *arXiv preprint arXiv:2102.07064*, 2021.

[59] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *NeurIPS*, 2020.

[60] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, *Implicit neural representations with periodic activation functions*, 2020. arXiv: 2006.09661 [cs.CV].

[61] J. Gao, W. Chen, T. Xiang, C. F. Tsang, A. Jacobson, M. McGuire, and S. Fidler, "Learning deformable tetrahedral meshes for 3d reconstruction," in *Advances In Neural Information Processing Systems*, 2020.

[62] M. Oechsle, S. Peng, and A. Geiger, *Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction*, 2021. arXiv: 2104.10078 [cs.CV].

[63] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Sep. 2019, pp. 5301–5310.

[64] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning," in *CVPR*, 2017.

[65] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Transactions on Graphics (TOG)*, 2019.

[66] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[67] A. Probst, D. Gatziolis, and N. Strigul, "Intercomparison of photogrammetry software for three-dimensional vegetation modelling," *Royal Society Open Science*, 2018.

[68] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 3, p. 29, 2013.

[69] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *ACM SIGGRAPH Computer Graphics*, 1987.

[70] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool," in *Eurographics Italian Chapter Conference*, V. Scarano, R. D. Chiara, and U. Erra, Eds., The Eurographics Association, 2008, ISBN: 978-3-905673-68-5.

[71] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, ser. SGP '06, Cagliari, Sardinia, Italy: Eurographics Association, 2006, pp. 61–70, ISBN: 3905673363.

[72] C. Quei-An, *Nerf_pl: A pytorch-lightning implementation of nerf*, 2020.

[73] L. Yen-Chen, *PyTorchNeRF: A PyTorch implementation of NeRF*, 2020.

[74] W. Xin and J. Pu, "An improved icp algorithm for point cloud registration," in *2010 International Conference on Computational and Information Sciences*, 2010.

[75] H. Men, B. Gebre, and K. Pochiraju, "Color point cloud registration with 4d icp algorithm," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1511–1516.

[76] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "Pointnetlk: Robust efficient point cloud registration using pointnet," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[77] F. Magistri, N. Chebrolu, and C. Stachniss, "Segmentation-based 4d registration of plants point clouds for phenotyping," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2433–2439.

[78] N. Chebrolu, T. Läbe, and C. Stachniss, "Spatio-temporal non-rigid registration of 3d point clouds of plants," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3112–3118.

[79] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3212–3217.

[80] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, *D-nerf: Neural radiance fields for dynamic scenes*, 2020. arXiv: 2011.13961 [cs.CV].

[81] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, "Deformable neural radiance fields," *arXiv preprint arXiv:2011.12948*, 2020.

[82] G. Gafni, J. Thies, M. Zollhöfer, and M. Nießner, *Dynamic neural radiance fields for monocular 4d facial avatar reconstruction*, 2020. arXiv: 2012.03065 [cs.CV].

[83] Z. Li, S. Niklaus, N. Snavely, and O. Wang, "Neural scene flow fields for space-time view synthesis of dynamic scenes," *arXiv preprint arXiv:2011.13084*, 2020.

[84] Y. Du, Y. Zhang, H.-X. Yu, J. B. Tenenbaum, and J. Wu, "Neural radiance flow for 4d view synthesis and video processing," *arXiv preprint arXiv:2012.09790*, 2020.